# STUDIO DELLA DIAGNOSI SUL DIABETE

COMPONENTI DEL GRUPPO:

Iosca Federica [MAT. 697611]: f.iosca2@studenti.uniba.it

A.A. 2023/2024

GitHub:

https://github.com/FedericaIosca/Progetto-ICON.git

# Sommario

# Introduzione al caso di studio

Il diabete è una malattia che si caratterizza per la presenza di quantità eccessive di glucosio (zucchero) nel sangue. L'eccesso di glucosio, noto con il termine di iperglicemia, può essere causato da un'insufficiente produzione di insulina o da una sua inadeguata azione; l'insulina è l'ormone che regola il livello di glucosio nel sangue. Le diagnosi di diabete stanno aumentando a livello mondiale e nel nostro Paese colpisce più di 3,5 milioni di persone ed è per questo motivo che è molto importante sapere qualcosa riguardo questa malattia.

La crescita del diabete nei Paesi occidentali è provocata soprattutto da fattori come l'invecchiamento della popolazione e le abitudini alimentari scorrette. Grazie però all'aumento delle diagnosi precoci si è abbassato il tasso di mortalità dei pazienti con diabete. Le forme più note di diabete sono due: il diabete di tipo 1 (con assenza di secrezione insulinica) e il diabete di tipo 2, conseguente a ridotta sensibilità all'insulina da parte di fegato, muscolo e tessuto adiposo e/o a una ridotta secrezione di insulina da parte del pancreas.

L'obbiettivo di questo progetto è analizzare e comprendere i vari sintomi che possono portare ad una diagnosi di diabete.

Il progetto è sviluppato in due modelli diversi:

1. Il primo contiene le descrizioni dei principali sintomi del diabete, e un test che attraverso le risposte dell'utente, fa una diagnosi;
2. Il secondo si basa sull'uso di algoritmi di apprendimento supervisionato.

# Requisiti funzionali

Il progetto è stato realizzato in Python in quanto offre un vasto numero di librerie che permettono di trattare e manipolare al meglio i dati in modo semplice.

IDE utilizzato: PyCharm Community Edition 2024.1. 5

# Dataset utilizzato

Il dataset, acquisito dalla piattaforma Kaggle (https://www.kaggle.com/uciml/pima-indians-diabetes-database), contiene informazioni relative a misurazioni cliniche di donne indiane di almeno 21 anni.

Il dataset contiene le seguenti informazioni:

- **Pregnancies**: numero di volte in cui è rimasta incinta
- **Glucose**: concentrazione plasmatica di glucosio a 2 ore in un test di tolleranza al glucosio orale
- **BloodPressure**: pressione diastolica (mmHg)
- **SkinThickness**: spessore della piega cutanea del tricipite (mm)
- **Insulin**: insulina sierica a 2 ore (mu U/ml)
- **BMI**: indice di massa corporea (peso in kg/(altezza in m)^2)

- **DiabetesPedigreeFunction**: funzione del pedigree del diabete
- **Age**: età
- **Outcome**: risultato che ha variabile 0 o 1 per capire se la donna in esame potrebbe avere il diabete o no

# Ontologie

In informatica, un'ontologia rappresenta formalmente e in modo condiviso una concezione esplicita di un determinato dominio di interesse. In termini più precisi, si tratta di una teoria assiomatica del primo ordine che può essere espressa in una logica descrittiva. Il concetto di ontologia formale è entrato in uso nel campo dell'intelligenza artificiale e della rappresentazione della conoscenza. Si riferisce alla maniera in cui vari schemi vengono combinati per creare una struttura dati che contiene tutte le entità rilevanti e le loro relazioni in un determinato dominio. I programmi informatici possono sfruttare l'ontologia per una serie di scopi, tra cui il ragionamento induttivo, la classificazione e diverse tecniche per la risoluzione di problemi.

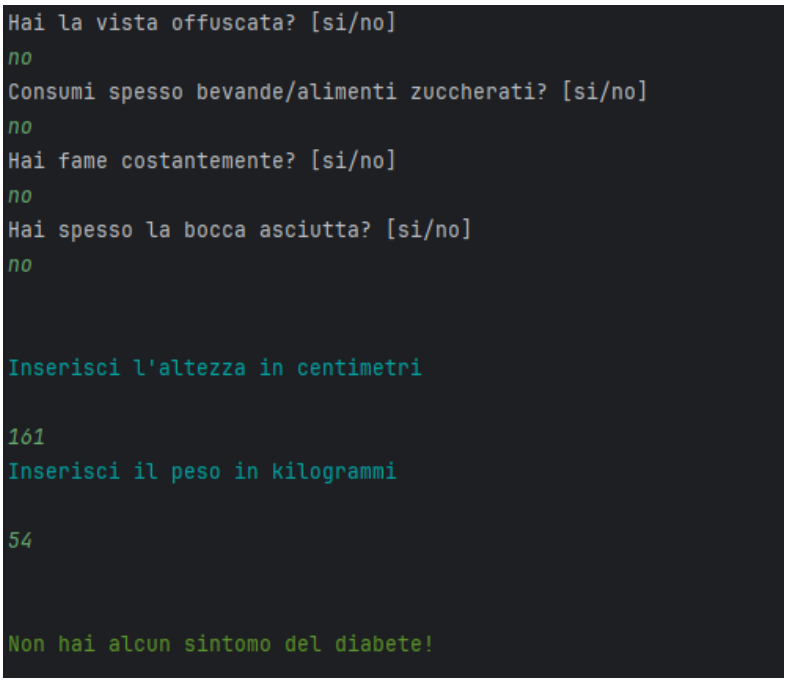
La lettura dell'ontologia, in questo caso, avviene tramite la libreria Python Owlready2 (come da figura sotto).

```
---------------->MENU<----------------

[1] Scopri i sintomi più comuni del diabete
[2] Esegui una diagnosi per scoprire se potresti soffrire di diabete di tipo 1
[3] Esci dal programma
1
Sintomo [1]: Nome: alimenti_zuccherati
Sintomo [2]: Nome: bocca_asciutta
Sintomo [3]: Nome: fame_costante
Sintomo [4]: Nome: perdita_peso
Sintomo [5]: Nome: prurito
Sintomo [6]: Nome: sete
Sintomo [7]: Nome: stanchezza
Sintomo [8]: Nome: vista_offuscata

Seleziona il sintomo di cui vuoi conosere la descrizione, inserisci il numero del sintomo
```

# Raccolta dati e ricerca

Durante la mia raccolta dati e ricerca sul diabete ho trovato vari test per fare una breve diagnosi sul diabete basando il tutto sui sintomi principali e più comuni che portano a diagnosticare il diabete di tipo 1. E per questo ho pensato di riprodurlo prendendo le più comuni domande. Cosi da indicare se l'utente che risponde alle domande ha la possibilità di avere il diabete di tipo 1 oppure no.

Il file è diabete.py.



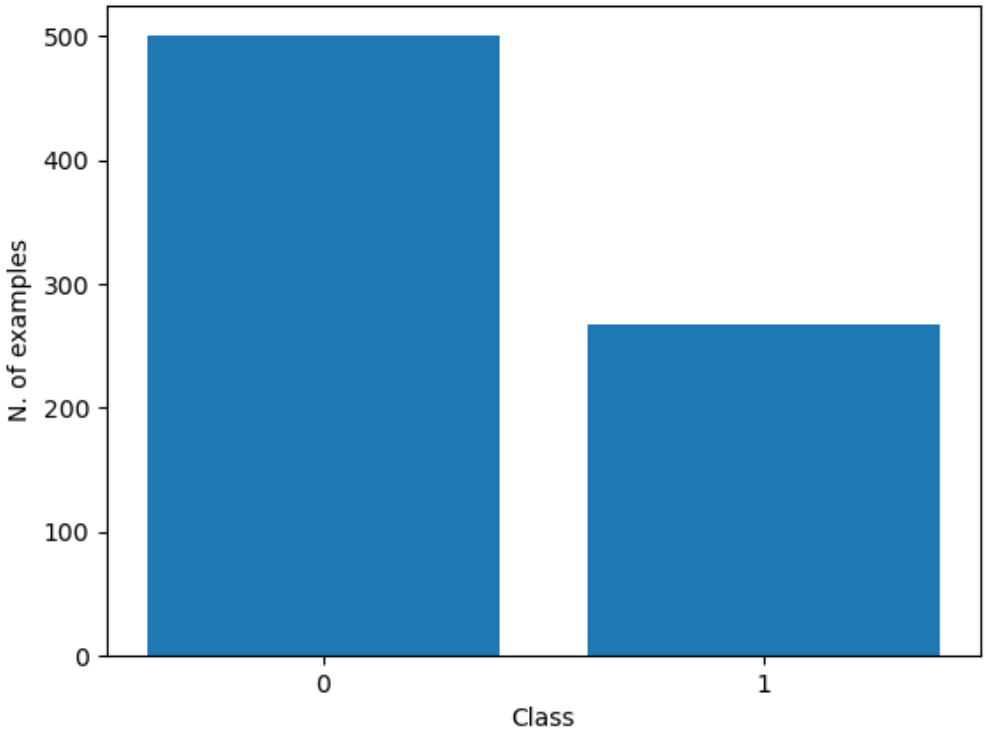

# Dataset

Questo dataset proviene originariamente dall'Istituto Nazionale per il Diabete e le Malattie Digestive e Renali. L'obiettivo del dataset `e predire diagnosticamente se un paziente ha il diabete, basandosi su determinate misurazioni diagnostiche incluse nel dataset. Sono state poste diverse restrizioni sulla selezione di queste istanze da un database più ampio. In particolare, tutte le pazienti qui presenti sono donne di almeno 21 anni di età di discendenza indiana Pima. Il dataset in questo progetto è il file "diabetes.csv" e all'interno possiamo trovare diverse variabili, di cui solo una variabile dipendente target, ovvero il campo Outcome.

## Analisi del dataset

### Controllo del bilanciamento del dataset

È importante controllare che i dati in un dataset siano bilanciati perchè avere un dataset bilanciato per un modello permetterebbe di ottenere modelli più accurati, una maggiore accuracy bilanciata e un tasso di rilevamento bilanciato più elevato. Di conseguenza, è fondamentale avere un dataset bilanciato per un modello di classificazione. Ciò significa che ogni classe viene considerata con la stessa importanza e può migliorare le prestazioni del modello. Se i dati sono sbilanciati, ciò può portare a risultati inaccurati e potrebbe essere necessario compiere ulteriori passaggi per bilanciare i dati prima dell'analisi. Per prima cosa, per condurre un'analisi di pre-processing accurata, verifichiamo se il dataset `e sbilanciato.




Come si può notare dall'immagine di sopra, il dataset non è per niente bilanciato. Ci sono più istanze della classe 0 che della classe 1.

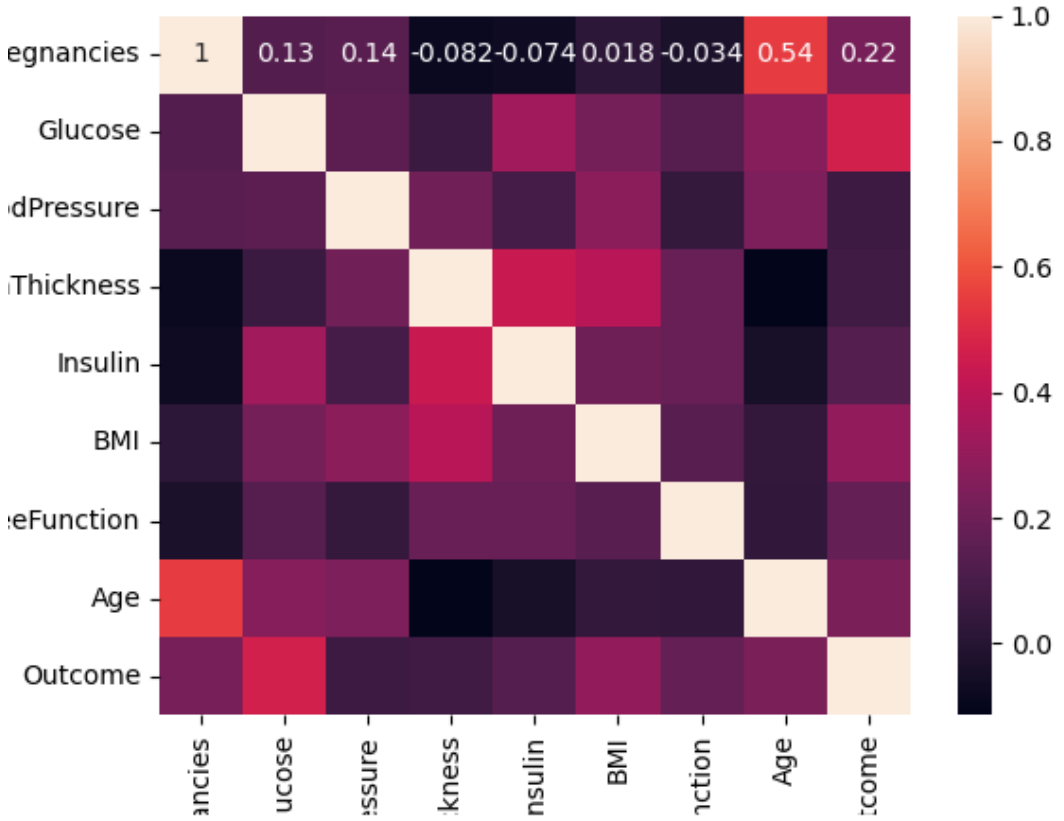
## Controllo di valori nulli

È giusto verificare anche la presenza di valori nulli nel dataset per individuare eventuali dati mancanti o incompleti. Questo può influenzare l'accuratezza dei risultati di un'analisi dei dati e potrebbe richiedere ulteriori fasi di pre-processing per gestire tali valori mancanti. In questo caso, non sono stati trovati valori nulli, quindi non è necessario eseguire alcuna operazione per risolvere questo problema.

```
Controllo valori nulli

Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

## Matrice di correlazione

Una matrice di correlazione è una tabella che evidenzia i coefficienti di correlazione tra le variabili. Ogni cella di questa tabella rappresenta la correlazione tra due variabili. Si tratta di uno strumento potente per riepilogare un insieme di dati e per individuare e visualizzare i modelli presenti nei dati. La matrice di correlazione in questo caso è la seguente:



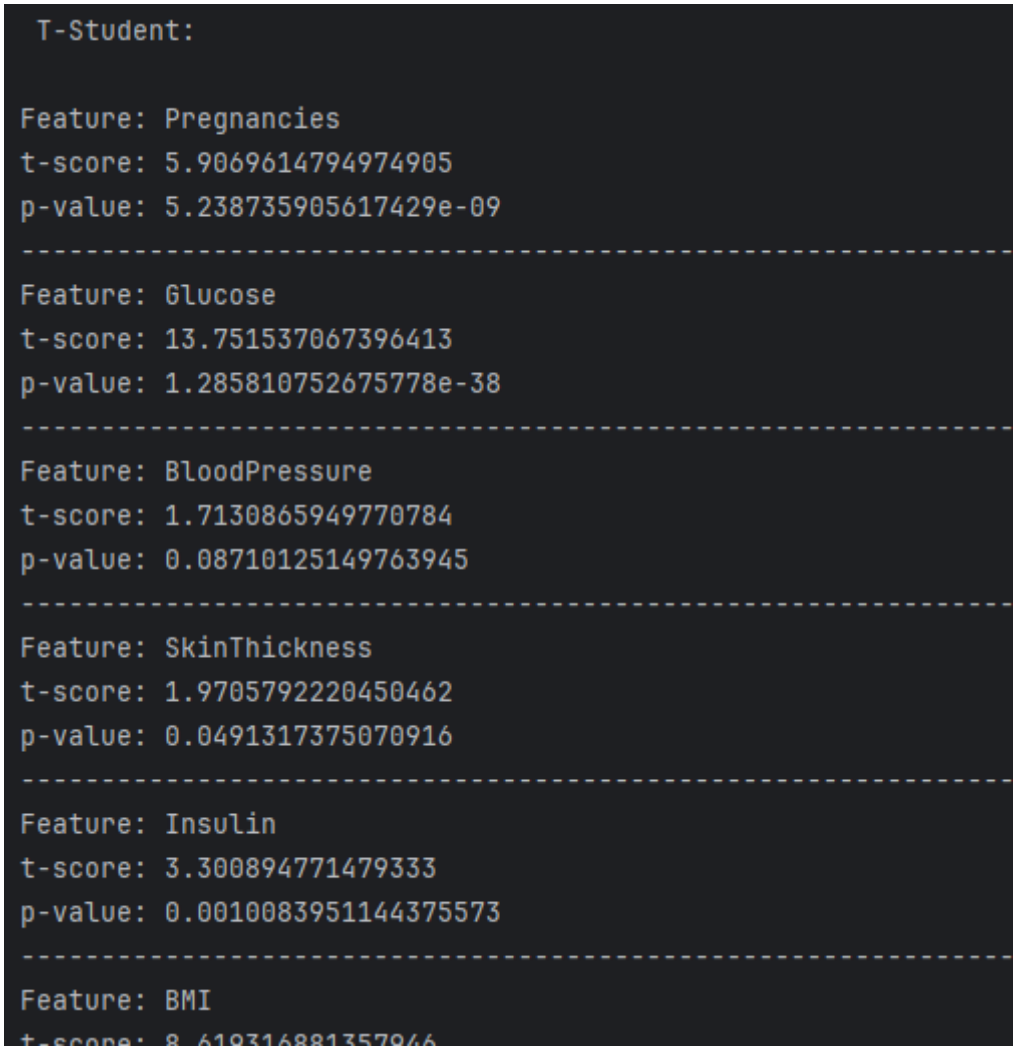

Da questa matrice di correlazione emerge che alcune caratteristiche presentano una correlazione più marcata tra loro rispetto ad altre. Ad esempio, Age e Pregnancies mostrano una correlazione positiva relativamente forte di 0,544341, mentre SkinThickness e Age evidenziano una correlazione negativa relativamente debole di -0,113970. La variabile Outcome, presumibilmente indicativa della presenza o meno del diabete in un individuo, mostra la correlazione positiva più significativa con il glucosio, pari a 0,466581. Purtroppo dal grafico sopra mostrato non è possibile vedere i dati precisi, quindi inserisco le seguenti immagini per avere più chiaro il tutto.

```
Matrice di correlazione

                          Pregnancies   Glucose  BloodPressure  SkinThickness  \
Pregnancies                  1.000000  0.129459       0.141282      -0.081672
Glucose                      0.129459  1.000000       0.152590       0.057328
BloodPressure                0.141282  0.152590       1.000000       0.207371
SkinThickness               -0.081672  0.057328       0.207371       1.000000
Insulin                     -0.073535  0.331357       0.088933       0.436783
BMI                          0.017683  0.221071       0.281805       0.392573
DiabetesPedigreeFunction    -0.033523  0.137337       0.041265       0.183928
Age                          0.544341  0.263514       0.239528      -0.113970
Outcome                      0.221898  0.466581       0.065068       0.074752

                           Insulin       BMI  DiabetesPedigreeFunction  \
Pregnancies              -0.073535  0.017683                 -0.033523
Glucose                   0.331357  0.221071                  0.137337
BloodPressure             0.088933  0.281805                  0.041265
SkinThickness             0.436783  0.392573                  0.183928
Insulin                   1.000000  0.197859                  0.185071
BMI                       0.197859  1.000000                  0.140647
DiabetesPedigreeFunction  0.185071  0.140647                  1.000000
Age                      -0.042163  0.036242                  0.033561
Outcome                   0.130548  0.292695                  0.173844
```

```
                                Age   Outcome
Pregnancies                0.544341  0.221898
Glucose                    0.263514  0.466581
BloodPressure              0.239528  0.065068
SkinThickness             -0.113970  0.074752
Insulin                   -0.042163  0.130548
BMI                        0.036242  0.292695
DiabetesPedigreeFunction   0.033561  0.173844
Age                        1.000000  0.238356
Outcome                    0.238356  1.000000
```

## T-student test

T-student test è un test statistico. L'ho scelto perché mi trovo di fronte a un problema di classificazione binaria, pertanto tale test può essere utile per stabilire se vi sia una differenza statisticamente significativa tra le medie di una variabile continua tra due gruppi (il gruppo con diabete e il gruppo senza diabete). Quindi calcoliamo se esiste una reale differenza significativa tra le medie di ciascuna variabile tra i due gruppi:




In effetti, come si può notare nell'immagine di sopra, i risultati mostrano i t-scores e i p-values del T-Student test per le variabili del dataset. Un t-score elevato e un p-value basso indicano che esiste una differenza statisticamente significativa tra le medie dei due gruppi per quella variabile. Ad esempio, per la variabile Glucose, il t-score è 13,751537067396413 e il p-value è 1,285810752675778e-38, a indicare che esiste una differenza molto significativa tra le medie dei due gruppi per questa variabile.

# Pre-elaborazione

Tenendo conto di quanto emerso nella parte di analisi del dataset, passiamo ora alla parte della pre-elaborazione.

Come abbiamo visto dalle analisi precedenti, nel dataset non ci sono valori nulli, ma ci sono valori errati impostati a 0. Quindi dobbiamo sostituire questi dati valori mancanti. Utilizziamo un oggetto SimpleImputer con la strategia di imputazione impostata su "mean" (media) e il valore mancante impostato su 0. Questo oggetto viene utilizzato per imputare i valori mancanti in tutte le colonne del dataset, tranne la colonna "Outcome". L'imputazione avviene richiamando il metodo fit transform dell'oggetto imputer sulle colonne rilevanti del dataset. Una volta imputati i valori mancanti, separiamo le capacità di input dalla classe di output eliminando la colonna "Outcome" dal dataset e memorizzandola in una variabile chiamata

features. La colonna “Outcome” viene memorizzata in una variabile separata chiamata labels. Successivamente aumentiamo la rilevanza delle feature significative trovate nell’analisi del dataset precedente, che sono: ’Pregnancies’, ’Glucose’, ’BMI’, ’DiabetesPedigreeFunction’, ’Age’.

## Normalizzazione dei dati

La normalizzazione è un passo importante nella fase di pre-processing o pre-elaborazione dei dati per l’apprendimento automatico, perché può contribuire a migliorare le prestazioni degli algoritmi. La normalizzazione trasforma i valori delle colonne numeriche del dataset in una common scale, senza distorcere le differenze negli intervalli di valori. Ciò può migliorare l’accuratezza dei risultati e ridurre la complessità temporale degli algoritmi. Inoltre, la normalizzazione può aiutare a prevenire l’overfitting e a migliorare la stabilità dell’addestramento del modello.

Quindi subito dopo, creiamo un oggetto MinMaxScaler e lo adattiamo alle feature di input richiamando il suo metodo fit. Le feature di input vengono quindi trasformate utilizzando lo scaler, richiamando il suo metodo transform e memorizzando il risultato in una variabile chiamata features normalized.

Validation set

Il validation set è un data set utilizzato per valutare la capacità di generalizzazione del modello durante l’addestramento. E viene utilizzato per ottimizzare gli iperparametri del modello. Si chiama validation set perchè serve a convalidare i risultati ottenuti nel training set. Se le prestazioni sono scarse, dobbiamo modificare gli iperparametri del modello e ricominciare l’addestramento fino a quando il risultato della validazione ci soddisfa. In questo modo, possiamo evitare il fenomeno dell’overfitting, in cui il modello è in grado di prevedere perfettamente i dati utilizzati nell’addestramento ma non è in grado di generalizzare su nuovi dati.

I dati vengono quindi suddivisi in training, validation e test set utilizzando la funzione di scikit-learn train test split. La dimensione del test set `e impostata a 0,3 (30%). Il training set `e il restante 70% che viene ulteriormente suddiviso in un training set e un validation set con una dimensione di validazione di 0,2 (30% del training set).

## Bilanciamento delle classi

Poiché già sappiamo che il dataset non è bilanciato, la ponderazione delle classi viene calcolata utilizzando la funzione compute class weight di scikit-learn con il peso delle classi impostato su ”balanced”, le classi impostate sulle unique labels delle classi in train labels e y impostata su train labels. I pesi delle classi risultanti sono memorizzati in un dizionario chiamato weight class dict.

## Salvataggio dei dati

Infine, tutti i dati vengono salbvati nella cartella Pre_Processed_Data utilizzando la funzione save della libreria numpy.

Nella cartella troviamo i seguenti file:

- class_weights.npy
- test_features.npy
- test_labels.npy
- train_features.npy
- train_labels.npy
- val_features.npy
- val_labels.npy

# Algoritmi di classificazione

Per questo progetto ho scelto di sperimentare 3 algoritmi di classificazione, ovvero:

- K-Nearest-Neighbor (KNN)
- Alberi di decisione (decision tree) e Random Forest

## K- Nearest-Neighbor

Il KNN è un modello di apprendimento supervisionato e non parametrico utilizzato per problemi di classificazione e regressione. L'obiettivo del KNN è fare previsioni per un nuovo esempio osservando i K punti più vicini in termini di distanza (ad esempio distanza euclidea).

Il modello KNN è capace di individuare relazioni non lineari tra le variabili.

## Oversampling

Per questo modello ho deciso di applicare SMOTE per bilanciare i dati, invece di prendere in considerazione i pesi della classe di minoranza calcolati nella parte di pre-elaborazione. SMOTE è l'acronimo di Synthetic Minority Oversampling Technique. Si tratta di una tecnica di oversampling per risolvere il problema dello squilibrio delle classi nei training set. Invece di duplicare semplicemente gli esempi della classe minoritaria, SMOTE genera nuovi esempi sintetici. SMOTE funziona selezionando esempi vicini nello spazio delle features, tracciando una linea tra gli esempi nello spazio delle features e disegnando un nuovo campione in un punto lungo quella linea. In particolare, viene prima scelto casualmente un esempio dalla classe di minoranza. Successivamente vengono individuati k dei vicini più prossimi per quell'esempio (tipicamente k=5). Viene quindi generato un nuovo esempio sintetico come combinazione lineare dell'esempio scelto e di uno dei suoi vicini più prossimi. Questa tecnica può essere molto efficace nel fornire informazioni aggiuntive al modello per aiutarlo ad apprendere il confine decisionale tra le classi. Quindi, SMOTE viene applicato al training set per bilanciare le classi. Ciò avviene generando nuove istanze sintetiche della classe minoritaria fino a quando il numero di istanze per ciascuna classe è uguale.

## Grid search

Utilizzo una grid search per trovare i migliori iperparametri per il modello sul validation set. La Grid Search è un metodo per selezionare i migliori iperparametri per un modello di machine learning. Per ogni iperparametro viene definita una griglia di valori possibili e il modello viene addestrato e valutato per ogni combinazione di iperparametri della griglia. La combinazione di iperparametri che produce le migliori prestazioni del modello viene selezionata come migliore. Nel codice, la griglia di iperparametri è definita per il parametro 'n_neighbors' del

classificatore k-NN e ogni combinazione di valori della griglia viene testata utilizzando il validation set per valutare le prestazioni del modello. La migliore combinazione di iperparametri trovata viene quindi utilizzata per addestrare il modello finale.

Nel codice accade questo:

1. Viene specificata una griglia di valori per l'iperparametro n del classificatore K-NN;
2. Viene inizializzata una variabile per tenere traccia dei migliori iperparametri e del miglior punteggio f1 trovati durante la ricerca della griglia;
3. Il codice esegue un loop su ogni valore di n neighbors nella griglia e addestra un modello k-NN con quel valore di n neighbors utilizzando il training set bilanciato. Le prestazioni del modello vengono valutate sull'insieme di validazione utilizzando il punteggio f1 come parametro;
4. Se l'f1-score del modello sul validation set ` e migliore dell'attuale f1-score trovato, i migliori iperparametri e il migliore f1-score vengono aggiornati con i valori attuali;
5. Infine, viene addestrato un modello k-NN finale utilizzando i migliori iperparametri trovati e l'insieme di allenamento bilanciato e le previsioni vengono fatte sull'insieme di test.

Ho scelto di utilizzare l'F1-score come metrica per valutare le prestazioni sul validation set perchè esso tiene conto sia dei falsi positivi (persone sane classificate come malate) sia dei falsi negativi (persone malate classificate come sane) e può fornire una visione più completa delle prestazioni del modello rispetto all'accuratezza, che potrebbe non essere la scelta migliore come metrica per trovare i migliori iperparametri poichè potrebbe essere influenzata dallo sbilanciamento delle classi.

## Decision Tree

Un Decision Tree è un tipo di algoritmo di apprendimento supervisionato utilizzato per task di classificazione e regressione. Ha una struttura gerarchica ad albero che consiste in un nodo radice, rami, nodi interni e nodi foglia.

Nel contesto della classificazione, un decision tree funziona dividendo ricorsivamente i dati in sottoinsiemi basati sui valori delle feature in ingresso. A ogni nodo interno dell'albero, viene eseguito un test su una delle feature in ingresso per determinare quale ramo seguire. Il test confronta il valore della feature con una soglia e invia i dati al nodo figlio sinistro o destro, a seconda che il valore sia inferiore o superiore alla soglia.

L'albero viene fatto crescere finchè tutti i dati di un nodo foglia non appartengono alla stessa classe o finchè non viene soddisfatto un criterio di arresto. Una volta cresciuto, l'albero può essere utilizzato per fare previsioni su nuovi dati partendo dal nodo radice e seguendo i rami fino a un nodo foglia in base ai valori delle feature di input. La label della classe al nodo foglia viene quindi restituita come predizione.

## Algoritmo genetico per trovare i migliori iperparametri

L'algoritmo genetico è un metodo di ottimizzazione ispirato alla selezione naturale e alla genetica. L'algoritmo genetico viene utilizzato per trovare i migliori iperparametri per il decision tree sul validation set.

L'algoritmo inizia creando una popolazione di individui, dove ogni individuo rappresenta un insieme di iperparametri. Gli iperparametri nel codice sono 'max_depth' e 'min_samples_split'. La popolazione viene quindi valutata utilizzando la funzione fitness, che in questo caso calcola l'F1-score del modello sul validation set utilizzando gli iperparametri correnti. Naturalmente, per selezionare gli iperparametri utilizziamo ancora una volta l'F1-score, poichè si tratta di un problema di classificazione binaria.

Successivamente, l'algoritmo esegue una serie di operazioni genetiche per creare una nuova generazione di individui. Queste operazioni comprendono la selezione (per scegliere gli individui più adatti alla riproduzione), il crossover (per scambiare informazioni genetiche tra due individui) e la mutazione (per introdurre variazioni casuali negli individui). Questo processo viene ripetuto per un numero predefinito di generazioni. Alla fine, l'algoritmo restituisce il miglior individuo trovato, che rappresenta l'insieme di iperparametri che massimizza la funzione fitness.

All'interno del codice, l'algoritmo genetico è implementato utilizzando la libreria DEAP. Gli operatori genetici (valutazione, crossover, mutazione e selezione) vengono registrati e l'algoritmo genetico viene eseguito utilizzando la funzione 'eaSimple' della libreria DEAP. Il metodo utilizzato nell'algoritmo genetico è la tournament selection. In questo metodo, un gruppo casuale di individui viene scelto dalla popolazione e l'individuo con il fitness-value più alto viene selezionato per la riproduzione. Questo processo viene ripetuto più volte per scegliere altri individui da allevare.

Nel codice viene utilizzato il crossover a due punti, che seleziona due punti casuali e scambia le informazioni genetiche tra i due per creare due nuovi individui. Viene utilizzata anche la mutazione uniforme, che cambia casualmente il valore di uno o più geni dell'individuo all'interno di un intervallo predefinito.

## Training del modello

Dopo aver trovato i migliori iperparametri con l'algoritmo genetico, ho istanziato un modello di decision tree passando come feature le classi pesate e i migliori parametri trovati con l'algoritmo genetico, ho addestrato il modello tramite la funzione fit e infine ho usato il modello addestrato per fare previsioni sul set test.

# Random Forest

Dopo il Decision Tree, ho deciso di costruire anche un Random Forest, utilizzando lo stesso approccio di ricerca degli iperparametri usato per il Decision Tree e, naturalmente, tenendo conto anche delle classi pesate.

Il random forest è un algoritmo di apprendimento supervisionato basato su alberi decisionali. È un modello di tipo ensemble, cioè combina le previsioni di più alberi decisionali per ottenere un risultato più accurato e robusto.

Il Random Forest consiste in una serie di classificatori (in questo caso, decision tree) e le loro previsioni vengono aggregate per identificare l'outcome più prevalente. Ciò può contribuire a migliorare le prestazioni del modello rispetto all'utilizzo di un singolo decision tree.

# Risultati del test set

Per valutare i risultati del test set, ho deciso di utilizzare le seguenti metriche:

- **Matrice di confusione**: La matrice di confusione è una tabella spesso utilizzata per descrivere le prestazioni di un modello di classificazione. Mostra il numero di true positive (TP), false positive (FP), true negative (TN) e false negative (FN) per ogni classe;
- **Accuracy**: è una misura delle prestazioni di un modello di classificazione. È definita come il rapporto tra il numero di previsioni corrette e il numero totale di previsioni. In altre parole, possiamo dire che misura quanti casi sono stati classificati correttamente dal modello;
- **Classification report**: Il Classification report fornisce una ripartizione dettagliata delle prestazioni dei modelli per ciascuna classe. Include diverse metriche come:
    - Precision: La precision è il rapporto tra i true positive e la somma dei true positive e dei false positive. Misura quante delle previsioni positive fatte dal modello sono effettivamente positive;
    - Recall: Il recall è il rapporto tra i true positive e la somma dei true positive e dei false negative. Misura quanti dei casi effettivamente positivi sono stati identificati correttamente dal modello;
    - f1-score: L'F1-score è la media armonica di precision e recall. Fornisce un'unica misura che bilancia sia la precision che il recall.
- **ROC**: questo è l'acronimo di Receiver Operating Characteristic. È un grafico che illustra le prestazioni di un classificatore binario al variare della sua soglia di discriminazione. La curva ROC viene creata tracciando il tasso di true positive (TPR) rispetto al tasso di false positive (FPR) a varie impostazioni di soglia;
- **AUC**: questo può essere interpretato come la probabilità che un caso positivo scelto a caso venga classificato dal classificatore come superiore a un caso negativo scelto a caso. Un'AUC elevata indica che il classificatore è in grado di discriminare efficacemente tra casi positivi e negativi;
- **Intervallo di confidenza**: l'intervallo di confidenza è uno strumento statistico utilizzato per stimare l'intervallo in cui si trova il valore vero di un parametro sconosciuto con un certo livello di confidenza. Può essere utilizzato per stimare l'intervallo in cui si trova il vero tasso di errore di un modello su un set di dati target. L'intervallo di confidenza è particolarmente utile quando si dispone solo di un piccolo test set, come nel caso di questo progetto, poichè fornisce una stima dell'incertezza associata alla stima del tasso di errore. In questo caso ho deciso di impostare l'intervallo di confidenza al 95%
- **K-fold cross validation**: Il K-fold cross validation sul training set è una a tecnica utilizzata per valutare le prestazioni di un modello e per mettere a punto i suoi

iperparametri. Si tratta di dividere i dati in k subsets, quindi di addestrare e valutare il modello k volte, utilizzando ogni volta un fold diverso con validation set.

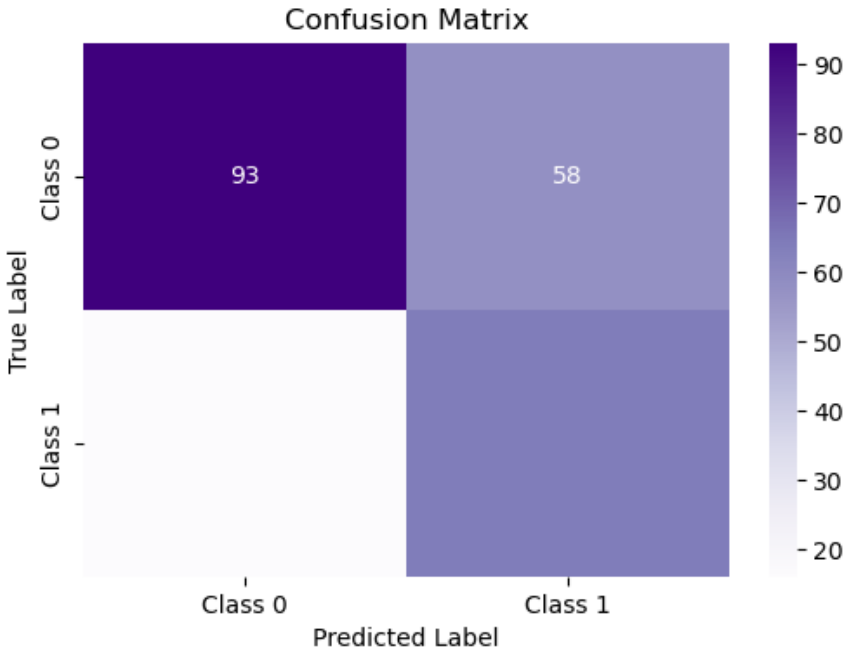
## Risultati sul KNN

L'accuracy sul test set è 0.68, questo significa che il 68% dei casi di test è stato classificato correttamente. L'intervallo di confidenza è [0.6170, 0.7365].

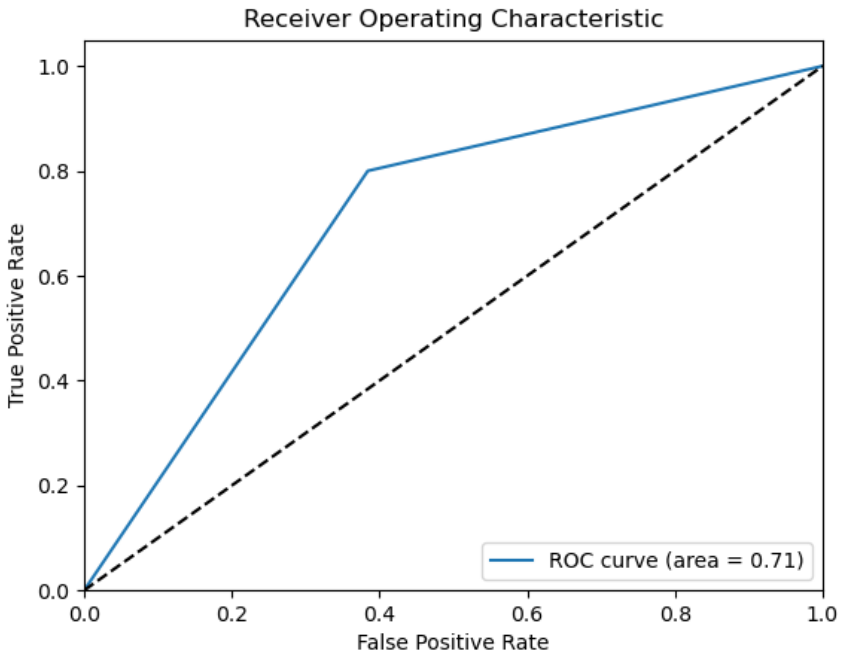
Il report di classificazione sottostante mostra come il KNN abbia una precision di 0.85 e una recall di 0.62 per la classe 0, e una precision di 0.52 e una recall di 0.80 per la classe 1. La precision per la classe 0 è di 0.85, questo significa che di tutti i casi che si prevedeva fossero di classe 0, solo l'85% erano di classe 0. La precision per la classe 1 è di 0.52, questo significa che di tutti i casi che si prevedeva fossero di classe 1, solo il 52% erano di classe 1. Il recall per la classe 0 è di 0.62, questo significa che di tutti i casi di classe 0, il 62% sono stati correttamente previsti come classe 0. Il recall per la classe 1 è di 0.80, questo significa che di tutti i casi previsti come classe 1, l'80% sono stati previsti come classe 1. L' f1-score per la classe 0 è di 0.72 e per la classe 1 è di 0.63.

```
Accuracy sul set di test: 0.68
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.62      0.72       151
           1       0.52      0.80      0.63        80

    accuracy                           0.68       231
   macro avg       0.69      0.71      0.67       231
weighted avg       0.74      0.68      0.69       231
```

La matrice di confusione (immagine sottostante), mostra come di 151 casi negativi, ovvero di classe 0, solo 93 sono stati correttamente predetti come reali negativi, e solo 58 sono stati erroneamente predetti come positivi, e quindi falsi positivi. Purtroppo i dati sulla classe 1 non sono riportati nel grafico, ma dai dati possiamo comunque intuire che su 80 casi positivi effettivi, ovvero classe 1, 64 sono stati correttamente predetti come positivi, cioè veri positivi e 16 sono stati erroneamente predetti come negativi, cioè falsi negativi.




Mentre dall'immagine sottostante possiamo notare come l'area sotto la curva ROC è 0.71, che indica che il modello ha una buona capacità di discriminazione.




Guarando ai risultati della k-fold cross validation sul training set, possiamo notare che in media, il classificatore knn classifica correttamente approssimativamente il 76.70% dei casi in ogni fold. L'accuratezza varia leggermente da fold a fold.

```
Intervallo di confidenza (95.0%): [0.6170, 0.7365]
Accuracy scores for each fold: [0.73255814 0.79069767 0.72093023 0.77906977 0.81176471]
Mean accuracy: 0.7670041039671682
```

Questi risultati ci fanno capire che il classificatore knn è in grado di ottenere prestazioni piuttosto buone su questo genere di problema.

# Risultati sul Decision Tree e Random Forest

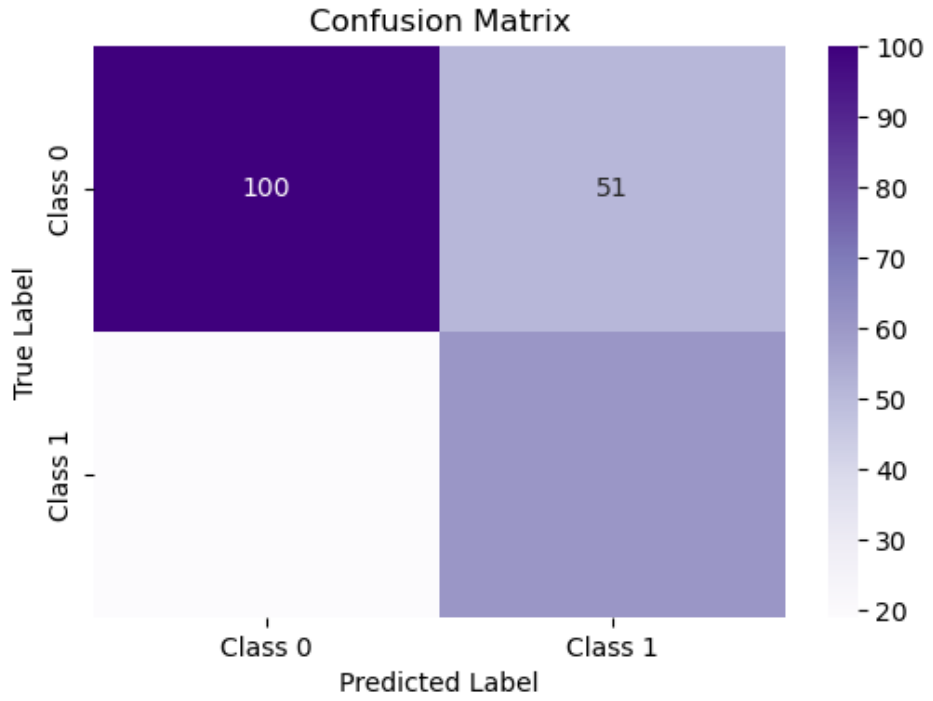
## Risultati sul Decision Tree

Dai risultati forniti, l'algoritmo del decision tree ha ottenuto un'accuracy sul test set pari a 70%, indicando che il modello è in grado di classificare correttamente il 70% delle istanze nel test set. L'intervallo di confidenza è [0, 6349, 0, 7526]. La matrice di confusione nell'immagine sottostante indica i seguenti risultati:

- 100 istanze sono state classificate correttamente come "0" (veri negativi)
- 51 istanze sono state classificate erroneamente come "1" (falsi positivi).

Anche se nel grafico non vengono mostrate dai calcoli fatti i risultati sono i seguenti:

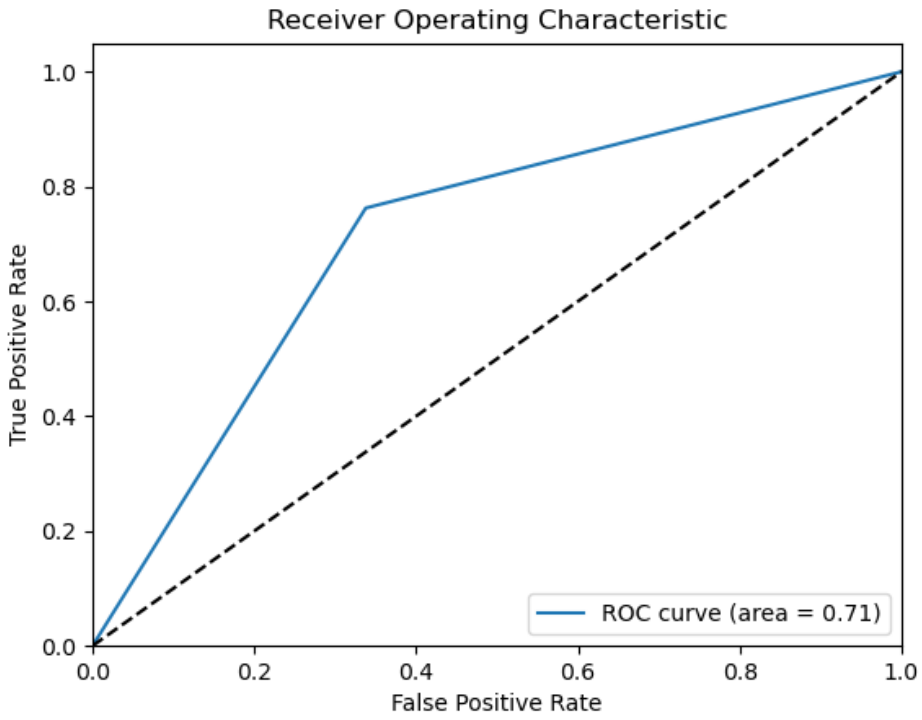
- 19 istanze sono state classificate erroneamente come "0" (falsi negativi).
- 61 istanze sono state classificate correttamente come "1" (veri positivi).




La precision nell'immagine sottostante per la classe "0" è di 84%, il che significa che di tutte le istanze classificate come "0", 84% di esse sono effettivamente "0". Il recall per la classe 0 è pari a 66%, il che indica che il 66% di tutte le istanze "0" presenti nel set di test sono state correttamente identificate come tali dal modello. Per la classe 1, la precision è di 54%, il che indica che di tutte le istanze classificate come "1", solo 54% di esse sono effettivamente "1". Il recall per la classe 1 è di 76%, il che significa che 76% di tutte le istanze "1" nel set di test sono state correttamente identificate come tali dal modello. L' f1-score per la classe 0 è 0,74, mentre per la classe 1 è 0,64. La media ponderata dell'accuracy è di 0,70, che tiene conto della distribuzione delle classi nell'insieme di test.

```
Accuracy sul set di test: 0.70
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.66      0.74       151
           1       0.54      0.76      0.64        80

    accuracy                           0.70       231
   macro avg       0.69      0.71      0.69       231
weighted avg       0.74      0.70      0.70       231
```

Mentre dall'immagine sottostante possiamo notare come l'area sotto la curva ROC è 0.71, indicando una buona capacità del modello di classificare correttamente le istanze positive rispetto a quelle negative.




Concludiamo dicendo che i risultati mostrano che il Decision Tree ha una performance buone nel classificare il diabete nel dataset utilizzato.

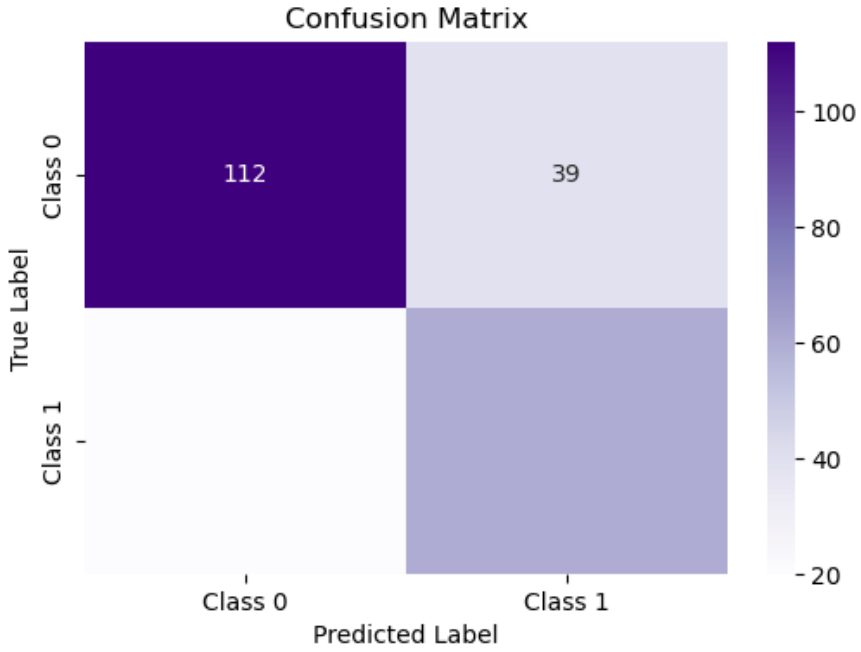
## Risultati del Random Forest

L'algoritmo Random Forest ha ottenuto un'accuratezza sul test set del 74%, leggermente superiore all'algoritmo Decision Tree. L'intervallo di confidenza ` e [0.6847, 0.7965]. La matrice di confusione nell'immagine sottostante mostra i seguenti risultati:

- 112 istanze sono state classificate correttamente come "0" (veri negativi)
- 39 istanze sono state classificate erroneamente come "1" (falsi positivi)

Anche se nel grafico non vengono mostrate dai calcoli fatti i risultati sono i seguenti:

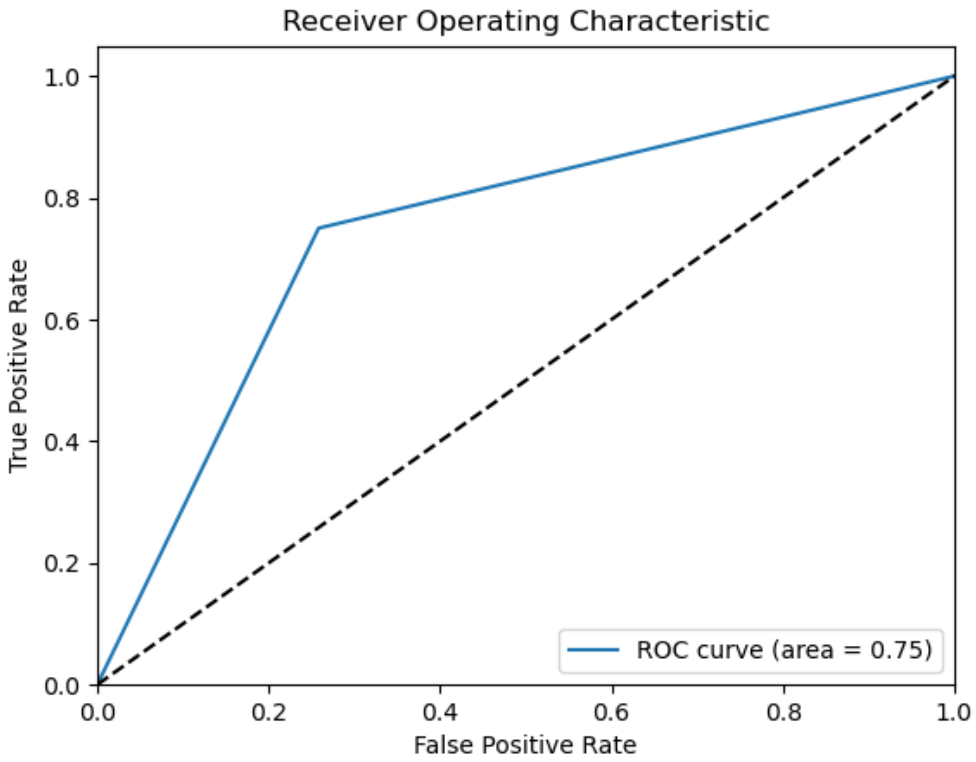
- 20 istanze sono state classificate erroneamente come ”0” (falsi negativi)
- 60 istanze sono state classificate correttamente come ”1” (veri positivi).




La precision nell’immagine sottostante per la classe 0 è pari a 85%, indicando che di tutte le istanze classificate come ”0”, 85% di esse sono effettivamente ”0”. Il recall per la classe 0 è di 74%, il che indica che 74% di tutte le istanze ”0” nel test set sono state correttamente identificate come tali dal modello. Per la classe 1, la precision è di 61%, a indicare che di tutte le istanze classificate come ”1”, solo 61% di esse sono effettivamente ”1”. Il recall per la classe 1 di 75%, il che significa che 75% di tutte le istanze ”1” nel test set sono state correttamente identificate come tali dal modello. L’F1-score per la classe 0 è di 0,79, mentre per la classe 1 è di 0,67. La media ponderata dell’accuratezza ` e di 0,74, che tiene conto della distribuzione delle classi nel test set.

```
Accuracy sul set di test: 0.74
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.74      0.79       151
           1       0.61      0.75      0.67        80

    accuracy                           0.74       231
   macro avg       0.73      0.75      0.73       231
weighted avg       0.76      0.74      0.75       231
```

Mentre dall'immagine sottostante possiamo notare come l'area sotto la curva ROC è 0.75, indicando una buona capacità del modello di classificare correttamente le istanze positive rispetto a quelle negative.




Dopo aver notato i risultati di entrambi, possiamo confermare che il Random Forest ha ottenuto risultati migliori rispetto al Decision Tree.

## Risultati della Cross-Validation

- **Decision Tree:**

```
Accuracy scores for each fold: [0.75581395 0.6744186  0.72093023 0.68604651 0.65882353]
Mean accuracy: 0.6992065663474691
```

- **Random Forest:**

```
Accuracy scores for each fold: [0.68604651 0.72093023 0.73255814 0.77906977 0.76470588]
Mean accuracy: 0.7366621067031464
```

Considerando l'accuracy media, si osserva che quella del Random Forest è superiore a quella del Decision Tree (0,7367 contro 0,6992). Ciò suggerisce che il Random Forest potrebbe avere una maggiore capacità di generalizzare e adattarsi ai dati. In effetti, il calcolo delle metriche conferma che il Random Forest è migliore del Decision Tree in questo contesto.

# Conclusioni generali

In generale, posso concludere che tutti e tre i modelli riescono a classificare in modo discreto tutti i dati del dataset. Ho fatto più fatica con le istanze della classe 1, ma probabilmente perchè il set di test è più popolato da istanze della classe 0 che da istanze di classe 1.