Ingegneria della Conoscenza
Diabete Prediction
Prof. Nicola Fanizzi

Gruppo di Lavoro
Notaro Lorenzo – 724931 – l.notaro1@studenti.uniba.it
Piergianni Giada – 717065 – g.piergianni4@studenti.uniba.it

Repository GitHub:
https://github.com/lnotaro1/IconDiabetePrediction-NotaroPiergianni.git

INTRODUZIONE

Questo caso di studio affronta diversi argomenti studiati nel corso di Ingegneria della conoscenza, in particolare il progetto affronta le seguenti tematiche:

- Applicazione dei modelli Logistic Regression, K-Nearest Neighbors, Decision Tree e Random Forest al dataset originale e al dataset bilanciato ottenuto applicando tre diverse tecniche di bilanciamento e confronto dei risultati ottenuti
- Applicazione del clustering sul dataset
- Applicazione del modello grafico probabilistico Bayesian Network, con l'obiettivo di calcolare la probabilità che un individuo, in base alle caratteristiche specificate, sviluppi la malattia.

Il progetto può fornire supporto medico sia per la diagnosi preventiva del diabete, aiutando gli specialisti a distinguere se un paziente ne sia affetto o meno, sia per identificare le caratteristiche condivise tra i pazienti affetti da tale disturbo. In questo modo, si può valutare quanto siano simili tra loro gli individui all'interno del campione di dati.

INFORMAZIONI TECNICHE

Il linguaggio di programmazione utilizzato è Python v. 3.11.6
Il software è Microsoft Visual Studio Code
Le librerie principali che si incontrano nel codice python sono:

- numpy →libreria che offre un supporto matematico e semplifica la gestione di calcoli matematici mediante array e funzioni matematiche
- pandas → libreria utilizzata per la gestione (lettura, pulizia, preparazione) di insiemi di dati strutturati in tabelle
- matplotlib → libreria usata per la visualizzazione dei dati python in grafici e personalizzazione degli stessi
- seaborn → libreria utilizzata per la visualizzazione di dati python (soprattutto dati statistici e categorici). Utilizzata spesso in combinazione con matplotlib per ottenere il massimo controllo e flessibilità nella creazione di grafici personalizzati
- sklearn → libreria utilizzata per l'apprendimento automatico. Essa fornisce strumenti efficaci per la modellazione statistica, la classificazione, la regressione, l'ottimizzazione dei parametri dei modelli
- pgmpy → libreria Python specializzata nella costruzione, l'addestramento e l'inferenza di modelli grafici probabilistici (PGM), che includono reti Bayesiane
- imblearn → è una libreria di Python progettata per affrontare il problema della classificazione su dataset sbilanciati; infatti, fornisce diverse tecniche di bilanciamento del dataset per gestire questa problematica.

ANALISI DEL DATASET

Il dataset preso in considerazione per il caso di studio raccoglie per ogni individuo le seguenti informazioni:

- gender
- age
- hypertension
- heart_disease
- smoking_history
- bmi
- HbA1c_level
- blood_glucose_level
- diabetes

In particolare:

- gender: si riferisce al sesso biologico dell'individuo
    - 0 = 'Female'
    - 1 = 'Male'
    - 2 = 'Other'
- age: età dell'individuo, può assumere valori compresi tra 0 e 80
- hypertension: l'ipertensione è una condizione medica in cui la pressione del sangue nelle arterie è persistentemente elevata
    - 0 = 'No'
    - 1 = 'Yes'
- heart_disease: si riferisce alla presenza di malattie cardiache
    - 0 = 'No'
    - 1 = 'Yes'
- smoking_history: si riferisce al rapporto che l'individuo ha con il fumo
    - 0 = 'No Info'
    - 1 = 'Never'
    - 2 = 'Former'
    - 3 = 'Current'
    - 4 = 'Not Current'
    - 5 = 'Ever'
- bmi: indice di massa corporea, misura del grasso corporeo basata sul peso e sull'altezza
- HbA1c_level: misura del livello medio di zucchero nel sangue di un individuo nell'arco degli ultimi 2-3 mesi.
- blood_glucose_level: si riferisce alla quantità di glucosio presente nel flusso sanguigno in un dato momento
- diabetes: variabile target, indica se una persona è affetta o meno da diabete
    - 0 = 'No Diabetes'
    - 1= 'Yes Diabetes'

Il dataset è formato da 100001 righe, e 9 colonne e tutti i valori sono di due tipi: interi o float; presenta 3854 elementi duplicati e nessun elemento nullo. Gli elementi duplicati sono stati successivamente eliminati. Quindi il dataset preso in considerazione presenta un totale di 96146 individui.

```
Number of null data:
 gender                0
age                    0
hypertension           0
heart_disease          0
smoking_history        0
bmi                    0
HbA1c_level            0
blood_glucose_level    0
diabetes               0
dtype: int64
Number of duplicated data:  3854
Number of duplicated data:  0
<class 'pandas.core.frame.DataFrame'>
Index: 96146 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   gender               96146 non-null  int64
 1   age                  96146 non-null  float64
 2   hypertension         96146 non-null  int64
 3   heart_disease        96146 non-null  int64
 4   smoking_history      96146 non-null  int64
 5   bmi                  96146 non-null  float64
 6   HbA1c_level          96146 non-null  float64
 7   blood_glucose_level  96146 non-null  int64
 8   diabetes             96146 non-null  int64
dtypes: float64(3), int64(6)
memory usage: 7.3 MB
```

Si possono distinguere le variabili categoriche da quelle numeriche. Le variabili categoriche sono: gender, hypertension, heart_disease, smoking_history e diabetes; le variabili numeriche sono: age, bmi, HbA1c_level e blood_glucose_level.

Per poter approfondire lo studio sul dataset è stata fatta un'esplorazione grafica dei dati.

ESPLORAZIONE GRAFICA DEI DATI

MATRICE DI CORRELAZIONE

Inizialmente si è visualizzata la matrice di correlazione

Una matrice di correlazione è una tabella che mostra i coefficienti di correlazione tra le variabili. Essa è composta da righe e colonne che mostrano le variabili. Ogni cella della tabella contiene il coefficiente di correlazione.

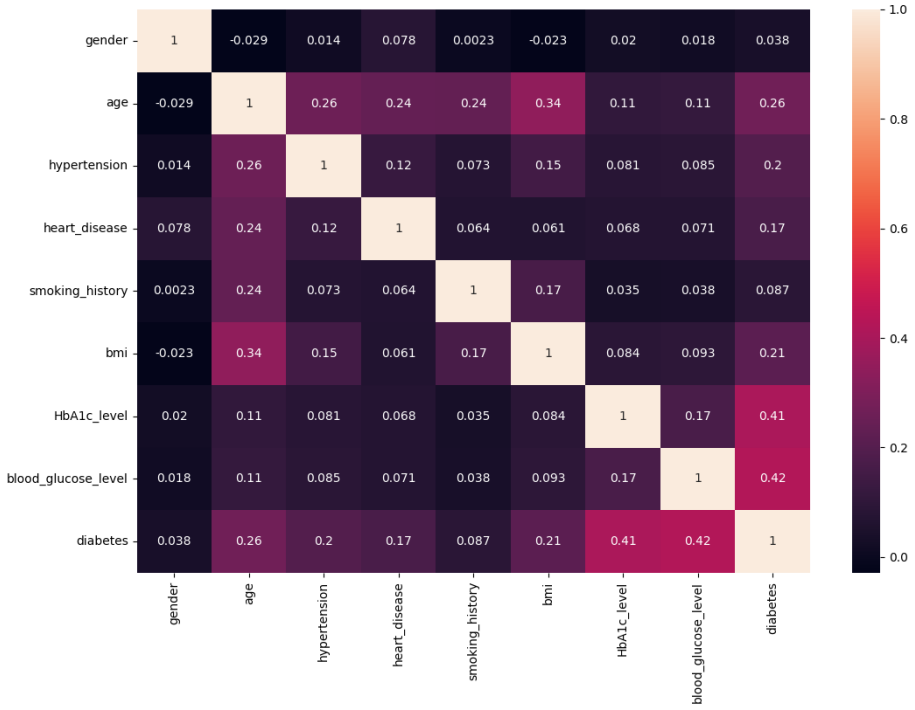
La correlazione è una misura che indica la relazione lineare fra due variabili casuali.

È compresa sempre fra -1 e 1, dove:

- -1 significa che le due variabili hanno una relazione lineare inversa, vale a dire che all'aumentare di una, l'altra diminuisce

- 1 significa che le due variabili hanno una relazione lineare diretta, vale a dire che all'aumentare di una aumenta anche l'altra

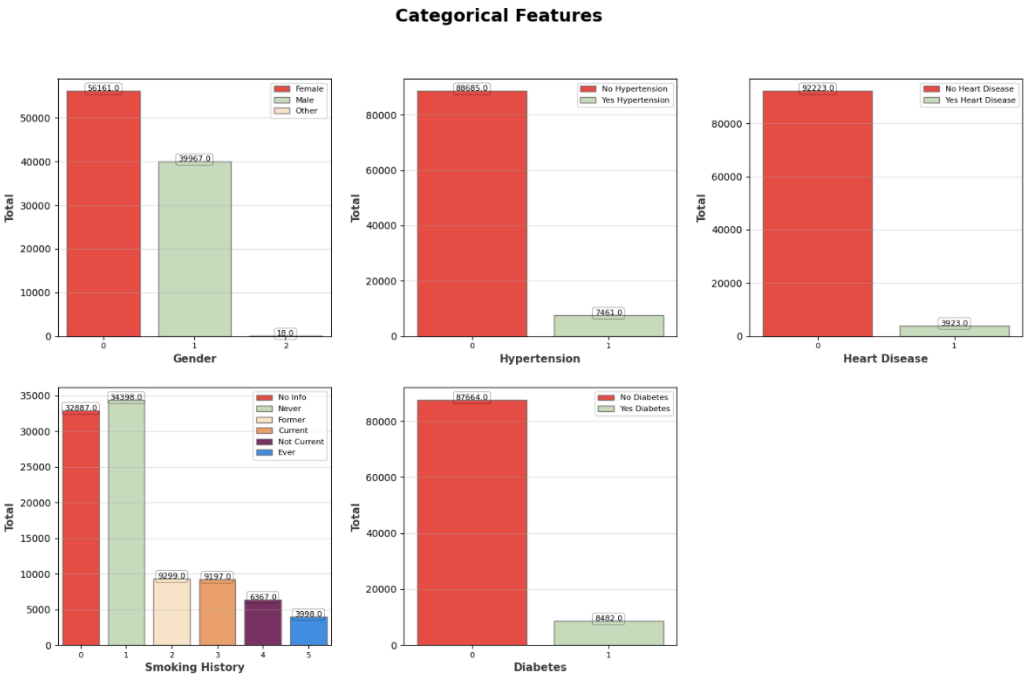
- 0 significa che non è possibile stabilire fra le due variabili un andamento lineare



Dalla matrice si può notare una bassa correlazione tra tutte le feature e quella target: diabetes. In particolare, si può osservare che le feature age e smoking_history sono quelle correlate meno, mentre hanno un indice di correlazione maggiore le feature HbA1c_level e blood_glucose_level.

VARIABILI CATEGORICHE

Si sono considerate le variabili categoriche singolarmente



gender: la maggior parte degli individui sono di sesso femminile

hypertension: sono più numerosi gli individui che non soffrono di ipertensione

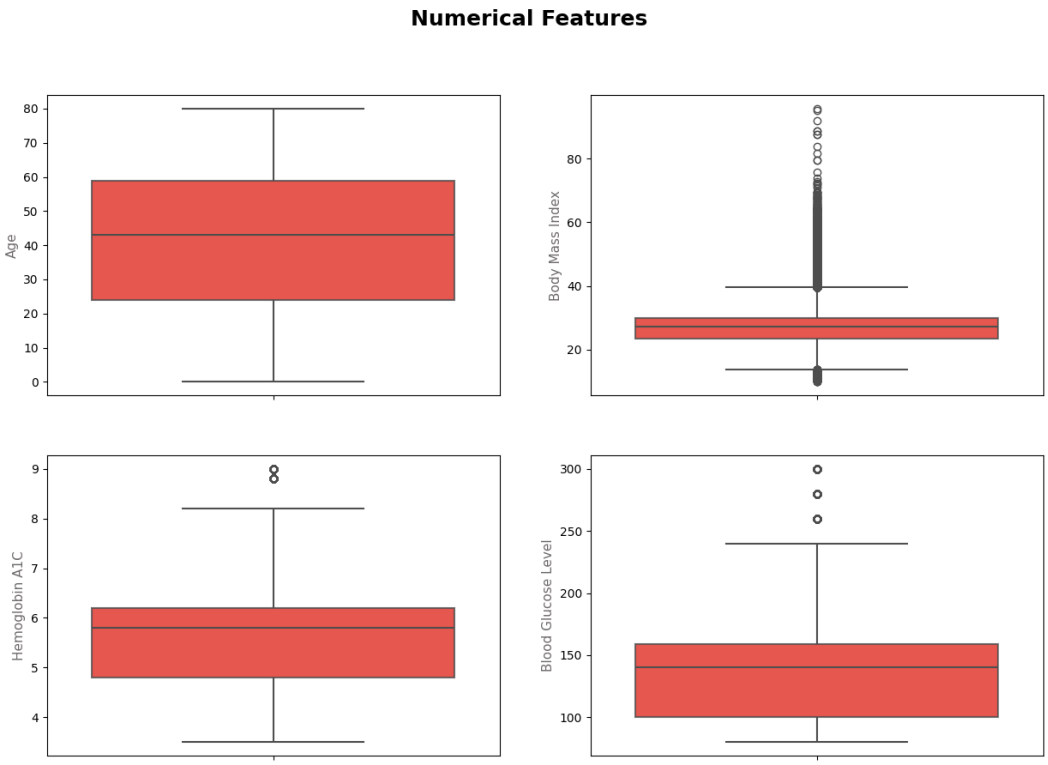
heart_disease: gli individui che presentano delle malattie cardiache sono numericamente inferiori rispetto a coloro che le presentano

smoking_history: la maggior parte degli individui presenti nel dataset non ha mai fumato o non si hanno informazioni

diabetes: si hanno pochi individui diabetici rispetto agli individui non diabetici

VARIABILI NUMERICHE

Si sono considerate le variabili numeriche singolarmente



age: la maggior parte degli individui ha un'età compresa tra i 25 e i 60 anni

bmi: i valori di indice di massa corporea sono compresi maggiormente tra 20 e 30

HbA1c_level: la maggior parte dei valori è compresa tra 5 e 6

blood_glucose_level: i valori sono compresi maggiormente tra 100 e 150

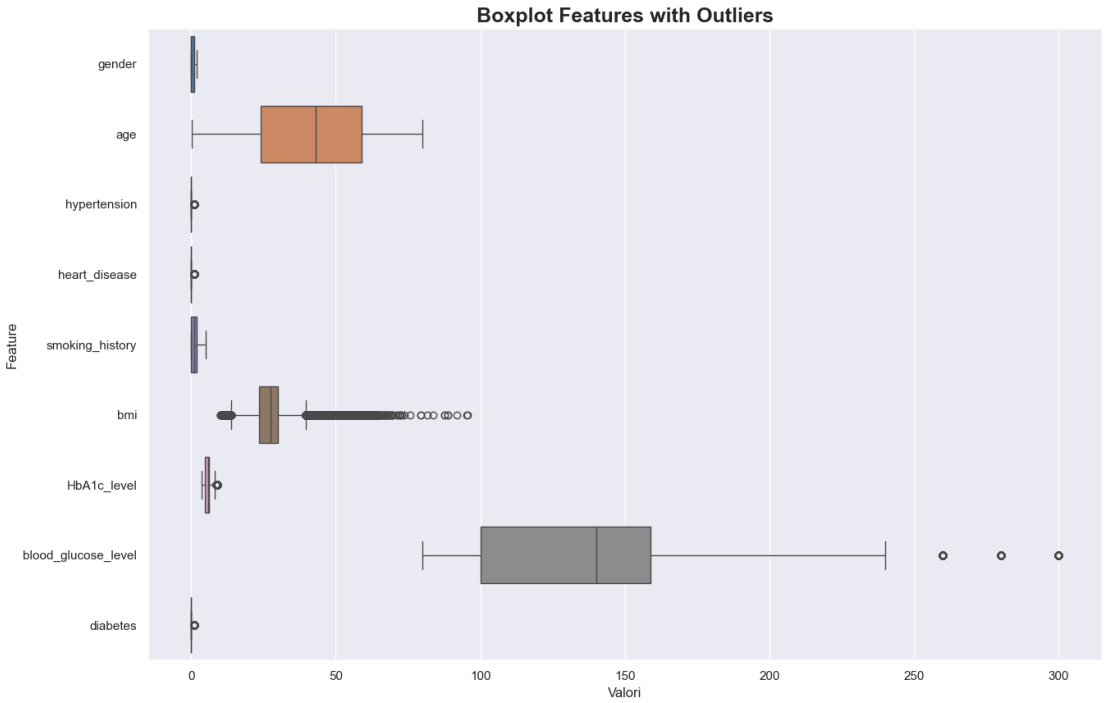
CONSIDERAZIONI SUL DATASET

L'esplorazione grafica dei dati permette di fare una considerazione fondamentale, ossia il dataset risulta essere fortemente sbilanciato.

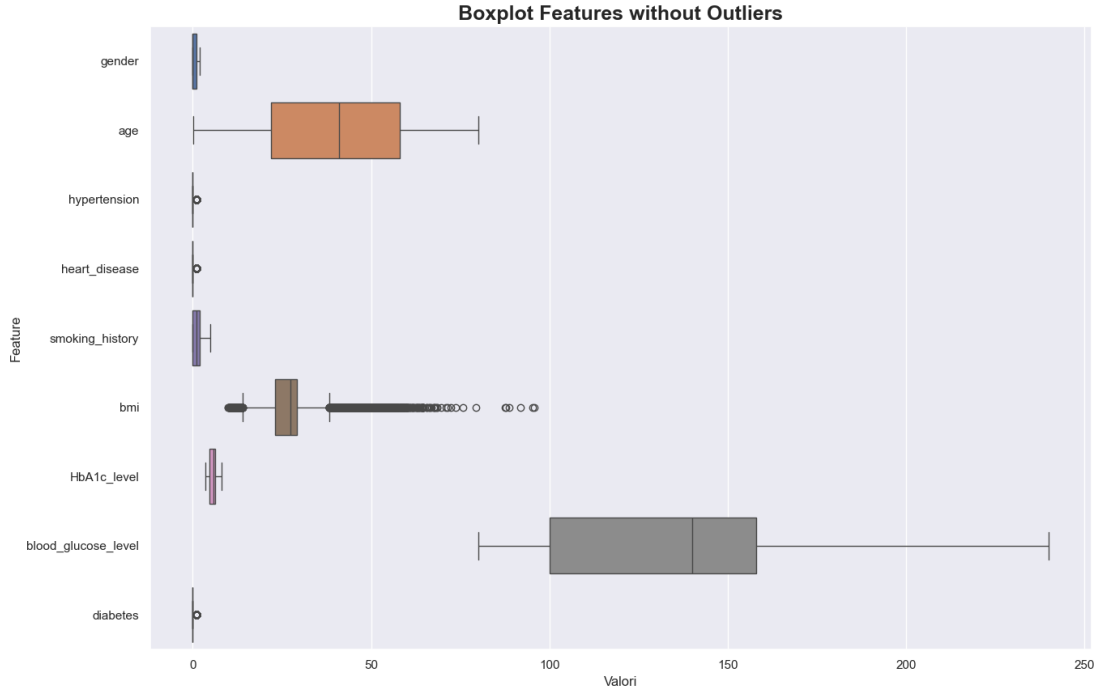
Si procede effettuando un'analisi degli outliers e la loro successiva eliminazione su un dataset di copia.

```
Total number of deleted outliers is: 8482
```

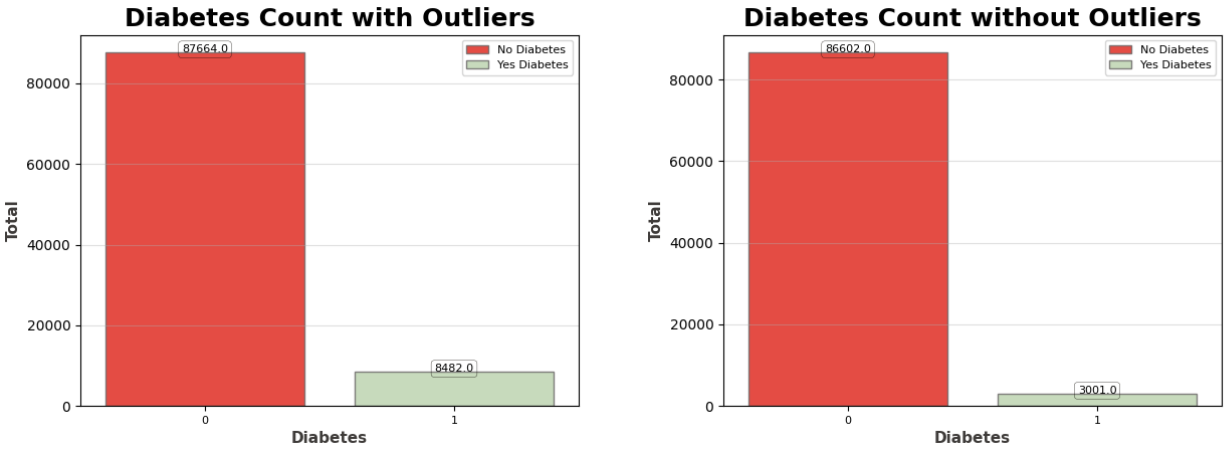
Questo è il boxplot ottenuto prima dell'eliminazione degli outliers



Questo è il boxplot ottenuto dopo l'eliminazione degli outliers



Tuttavia, se prendiamo in considerazione la nostra variabile target prima e dopo l'eliminazione degli outliers otteniamo questo:



Si può notare che sono rimasti solo 3001 diabetici nell'intero dataset. Si può concludere quindi che eliminando gli outliers accentuiamo ulteriormente lo sbilanciamento delle due classi; quindi, si è deciso di non procedere con l'eliminazione utilizzando il dataset originale.

I risultati ottenuti però sono stati presi in considerazione, si è infatti deciso successivamente di utilizzare il Robust Scaler come metodo di scaling dei dati, perché questo non è influenzato dagli outliers.

SCELTE DI PROGETTO

FEATURE SCALING

Prima di avanzare con l'analisi, è stato deciso di applicare una standardizzazione ai dati utilizzando il RobustScaler. Questo perché è un metodo di scaling che si avvale della mediana e del quartile per determinare la scala dei dati, rendendolo meno influenzabile dalla presenza degli outlier o valori fuori scala. Questa caratteristica lo rende particolarmente adatto quando i dati contengono numerosi valori anomali che potrebbero influenzare negativamente le performance di alcuni algoritmi di machine learning.

TRAIN TEST SPLITTING STRATIFICATO

Avendo un dataset fortemente sbilanciato si è deciso di applicare un train test splitting stratificato per suddividere il dataset in un set di train e un set di test in modo che le proporzioni delle classi siano mantenute in entrambi i set.

ADDESTRAMENTO DEI MODELLI CON CROSS VALIDATION

Nel caso di studio si è deciso di utilizzare la stratified k-fold cross-validation che garantisce che ogni fold abbia una distribuzione delle classi simile a quella del dataset complessivo. In altre parole, la stratificazione preserva la proporzione delle classi in ciascun fold.

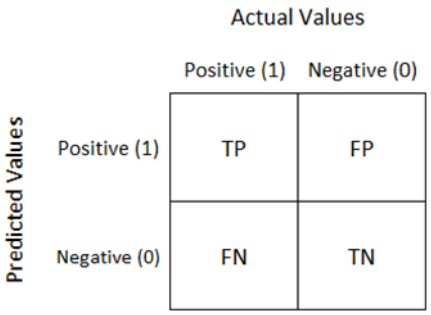
Inizialmente si è proceduto con il training dei modelli di apprendimento supervisionato utilizzati, ottimizzando il valore di recall. La recall è una metrica che ci permette di minimizzare i falsi negativi, e quindi massimizzare la capacità del modello di individuare

correttamente gli esempi positivi, ossia coloro che sono individuati dal modello come diabetici e che effettivamente presentano la malattia.

TUNING DEGLI IPERPARAMETRI

Per ciascun modello si è effettuato il tuning degli iperparametri con GridSearchCv, ottimizzando la recall e anche in questo caso si è utilizzata la cross validation. Sono stati poi calcolati i valori di accuracy, precision, recall e f1 e sono state visualizzate la matrice di confusione e la curva ROC.

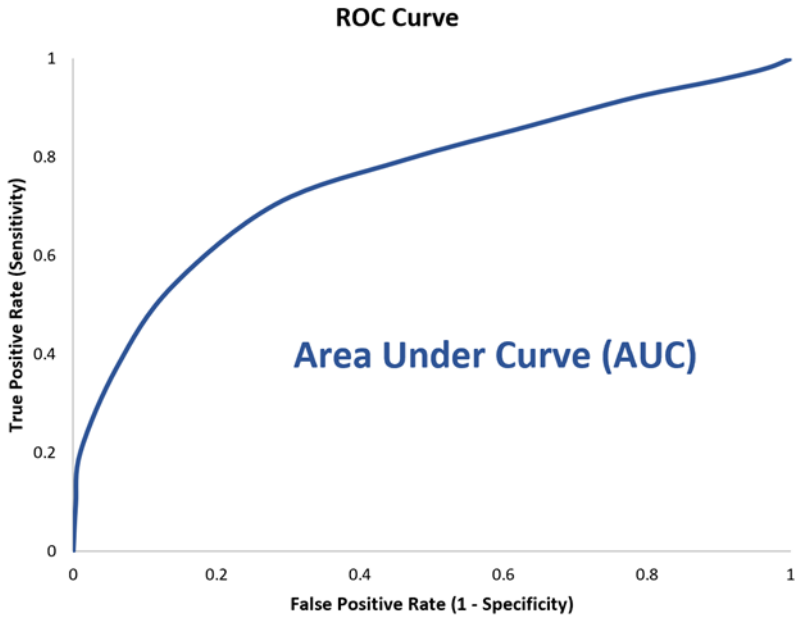
La matrice di confusione è una tabella utilizzata nella classificazione dei modelli di machine learning per valutare le prestazioni di un algoritmo di classificazione. Essa mostra la relazione tra le previsioni del modello e le etichette reali dei dati, è composta da quattro parti principali: True Positives (previsioni corrette di classi positive), True Negatives (previsioni corrette di classi negative), False Positives (previsioni errate di classi positive) e False Negatives (previsioni errate di classi negative).



La ROC è una curva di probabilità e l'AUC è una misura numerica utilizzata per valutare le prestazioni di un classificatore binario basato sulla curva ROC. L'AUC rappresenta l'area sottesa dalla curva ROC ed è comunemente utilizzato come indicatore di quanto bene un classificatore possa discriminare tra le classi positive e negative. Maggiore è l'AUC, migliore è la capacità predittiva del modello.

Sull'asse delle ordinate della curva ROC, si rappresenta il True Positive Rate, TPR, che è il numero di campioni positivi correttamente classificati rispetto al totale dei campioni positivi; sull'asse delle ascisse, si rappresenta il False Positive Rate, FPR, che è il numero di campioni negativi erroneamente classificati come positivi rispetto al totale dei campioni negativi.

L'AUC è un valore compreso tra 0 e 1. Un valore pari 0 indica che il classificatore è completamente inadeguato e non è in grado di discriminare tra le classi. Un valore pari a 1 indica che il classificatore è perfetto e può separare completamente le classi senza errori. Quando l'AUC è pari a 0,7, significa che c'è il 70% di possibilità che il modello sia in grado di distinguere tra classe positiva e classe negativa. Quando l'AUC è circa 0,5, il modello non ha alcuna capacità di discriminazione.



BILANCIAMENTO DEL DATASET

Dato l'evidente sbilanciamento del dataset si è deciso di bilanciarlo utilizzando tre metodi:

1. applicazione di pesi alle classi
2. applicazione di random oversampling con SMOTE
3. applicazione di random undersampling.

Per quanto riguarda l'applicazione dei pesi alle classi, questo si è ottenuto andando a specificare per ciascun modello il valore per il parametro 'class_weight'; per il K-Nearest Neighbors, invece, è stato utilizzato il parametro 'weights' impostato come 'uniform', questo permette di assegnare lo stesso peso a tutti i vicini di un punto.

Per quanto riguarda invece gli ultimi due metodi le tecniche mirano a risolvere il problema di classi sbilanciate attraverso approcci diversi. Il Random Oversampling cerca di aumentare la rappresentanza della classe di minoranza, mentre il Random Undersampling cerca di ridurre la rappresentanza della classe di maggioranza.

APPRENDIMENTO SUPERVISIONATO

L'apprendimento supervisionato è una tecnica di apprendimento automatico che si concentra sull'addestramento di un sistema in modo da consentirgli di effettuare previsioni o classificazioni automatiche in risposta a determinati input. Queste previsioni o classificazioni sono basate su un insieme di dati di addestramento costituito da esempi che contengono sia gli input che i corrispondenti output desiderati. Questi esempi di addestramento sono forniti al sistema inizialmente. L'obiettivo dell'apprendimento supervisionato è far sì che il sistema possa fare previsioni accurate su nuovi dati o input che non sono stati precedentemente osservati.

Si parla di apprendimento supervisionato perché si hanno dati etichettati, ossia un insieme di dati in cui ogni esempio o campione di dati è associato a un'etichetta o a una categoria che indica l'output desiderato o la classe a cui appartiene. Queste etichette forniscono informazioni sul "corretto" risultato o sulla categoria a cui dovrebbe appartenere ciascun esempio di dati.

L'apprendimento supervisionato può essere distinto in due tipi di problemi: classificazione e regressione.

La differenza principale tra regressione e classificazione risiede nell'output previsto. La regressione prevede valori numerici continui, mentre la classificazione prevede categorie discrete.

La classificazione utilizza un algoritmo per assegnare accuratamente i dati di test a categorie specifiche. Riconosce delle entità specifiche all'interno del set di dati e cerca di trarre delle conclusioni su come queste entità dovrebbero essere etichettate o definite. La classificazione è utilizzata quando l'obiettivo è assegnare una categoria o una classe a ciascun esempio di input.

La regressione viene utilizzata per comprendere la relazione tra variabili dipendenti e indipendenti. La regressione è utilizzata quando l'obiettivo è prevedere un valore numerico continuo come output.

In questo caso di studio sono stati utilizzati quattro algoritmi di apprendimento supervisionato:

- Logistic Regression
- K-Nearest Neighbors
- Decision Tree
- Random Forest

LOGISTIC REGRESSION

La regressione logistica è un metodo statistico utilizzato per costruire modelli di machine learning in cui la variabile dipendente è dicotomica: cioè binaria. La regressione logistica è chiamata così perché utilizza la funzione logit (log-odds) per modellare la probabilità che un'osservazione appartenga a una delle due categorie della variabile dipendente. Questo modello appiattisce la funzione di regressione lineare utilizzando la funzione sigmoide.

Funzione della regressione lineare:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots.. + \beta_n X_n$$

Funzione sigmoide:

$$p = \frac{1}{1 + e^{-y}}$$

K-NEAREST NEIGHBORS

Il K-nearest neighbor, noto anche come algoritmo KNN, è un algoritmo che classifica i punti dati in base alla loro vicinanza e associazione con altri dati disponibili. Questo algoritmo presuppone che punti dati simili possano essere trovati uno vicino all'altro, 'k' indica appunto il numero di punti più vicini a quello da classificare, ed è un intero positivo. Inizialmente avviene la definizione del parametro k, poi viene calcolata la distanza tra punti dati. Quelle più comuni sono la distanza euclidea o la distanza di Manhattan e infine assegna una categoria basandosi sulla categoria più frequente.

DECISION TREE

Il decision tree è un modello di apprendimento automatico utilizzato per la classificazione e la regressione. L'obiettivo di un albero decisionale è suddividere un set di dati basandosi su regole decisionali derivate dai dati stessi. Questo processo crea una struttura a forma di albero dove i nodi foglia rappresentano le classi per problemi di classificazione o valori numerici per problemi di regressione. I nodi foglia sono etichettati con la classe o il valore previsti.

RANDOM FOREST

Random Forest è un algoritmo di machine learning basato su alberi che sfrutta la potenza di più alberi decisionali per prendere decisioni. L'algoritmo costruisce un gran numero di alberi decisionali individuali, su diversi sottoinsiemi del dataset originale, che operano come un insieme. Ogni singolo albero nel Random Forest fa una previsione di classe e la classe che è stata predetta maggiormente diventa il risultato che dà il modello.

Ogni algoritmo di apprendimento è stato valutato prima e dopo il bilanciamento del dataset.

APPLICAZIONE DELLE SCELTE DI PROGETTO

Inizialmente si è effettuato l'addestramento dei modelli con ottimizzazione del valore di recall e si sono ottenuti seguenti risultati:

```
Logistic Regression
Recall Scores:  [0.66498316 0.6026936  0.62257793 0.61836563 0.64785173]
Average Recall Score :  0.6312944099801724


K-Nearest Neighbors
Recall Scores:  [0.58670034 0.56060606 0.56697557 0.56613311 0.60572873]
Average Recall Score :  0.5772287605059299

Decision Tree
Recall Scores:  [0.76515152 0.72306397 0.73546757 0.75231676 0.74978939]
Average Recall Score :  0.7451578406928029

Random Forest
Recall Scores:  [0.71969697 0.66835017 0.67228307 0.68660489 0.70345409]
Average Recall Score :  0.6900778353600594
```

APPLICAZIONE DEI MODELLI AL DATASET ORIGINALE

TUNING DEGLI IPERPARAMETRI

Successivamente si è proceduto con il tuning degli iperparametri per tutti i modelli.

Per l'ottimizzazione degli iperparametri del modello Logistic Regression sono stati scelti i seguenti valori:

```
param_grid_log_reg = [
    {'C' : [0.001, 0.01, 0.1, 1, 10, 100, 1000],'penalty' : ['l2'],'max_iter' : [100,1000,5000]},
    {'C' : [0.001, 0.01, 0.1, 1, 10, 100, 1000], 'penalty' : ['l1'],'max_iter' : [100,1000,5000]}]
```

Solver: algoritmo utilizzato nei problemi di ottimizzazione (default = 'lbfgs')

C: parametro di regolarizzazione che controlla la forza della regolarizzazione (default = 1.0)

Penalty: parametro che specifica il tipo di regolarizzazione da utilizzare (default = l2)

'l1': ha una "funzione di selezione" incorporata che aiuta a selezionare le feature più rilevanti per il modello, riducendo il numero di feature utilizzate

'l2': non ha una funzione di selezione incorporata come L1, ma riduce l'ampiezza dei coefficienti, limitando così l'overfitting

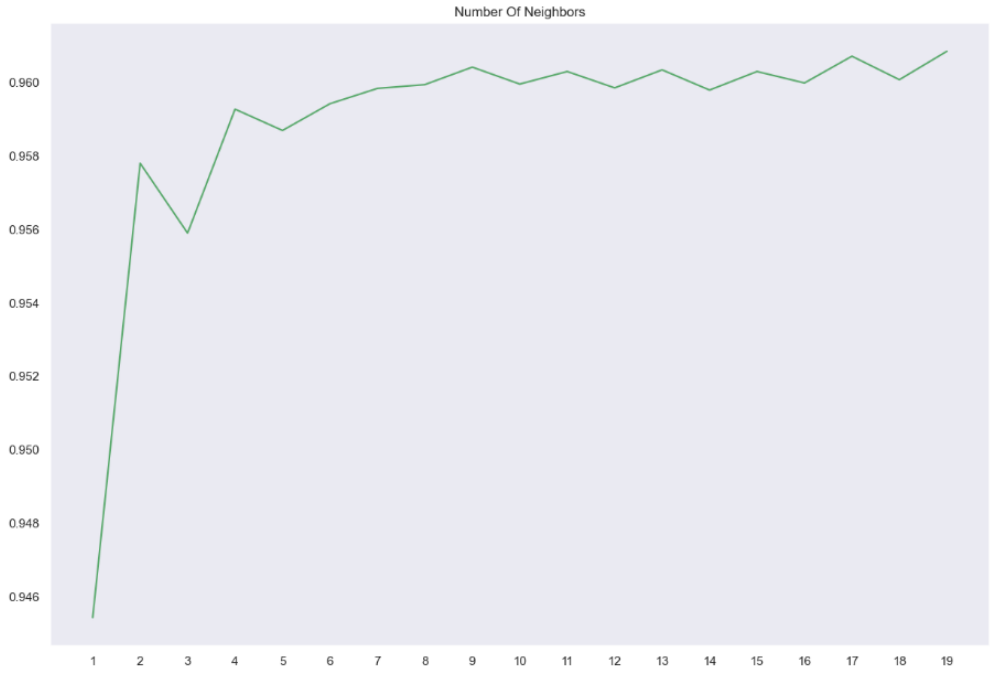
Max_iter: parametro che definisce il numero massimo di iterazioni per la convergenza (default = 100)

In questo caso il solver utilizzato è il 'liblinear' perché è il più indicato per dataset relativamente piccoli, come quello scelto in questo caso di studio. Il solver 'liblinear' ha condizionato la scelta dei due valori per penalty: l1, l2.

Al fine di selezionare i migliori iperparametri è stato utilizzato il GridSearchCV() e poi si sono calcolati i valori di accuracy, recall, precision e f1

```
Best Parameters Logistic Regression: {'C': 10, 'max_iter': 100, 'penalty': 'l2'}
                 Model  Accuracy  Precision    Recall  F1 Score
0  Logistic Regression  0.958397   0.864499  0.626719  0.726651
```

Per il modello del K-Nearest Neighbors il primo passo è la definizione del parametro k e in questo caso specifico il k è stato scelto applicando la 5-fold cross-validation



Si è ottenuto k = 19.

Per l'ottimizzazione degli iperparametri del modello sono stati scelti i seguenti valori:

```
param_grid = {
    'weights':['uniform','distance'],
    'algorithm':['auto','ball_tree','kd_tree','brute'],
    'leaf_size':[10,20,30,40,50],
    'p':[1,2],
    'metric': ['euclidean','manhattan','minkowski'],
}
```

Weights: specifica il tipo di peso da assegnare ai vicini durante il calcolo delle predizioni (default ='uniform')

- 'uniform': pesi uniformi. Tutti i vicini hanno lo stesso peso.
- 'distance': pesa i punti in base all'inverso della loro distanza. In questo caso, i vicini più prossimi avranno un'influenza maggiore rispetto ai vicini più lontani.

Algorithm: specifica l'algoritmo da utilizzare per trovare i vicini più vicini (default = ’auto’)

- 'auto' : cercherà di selezionare l'algoritmo più adatto in modo automatico, in base al dataset e agli altri iperparametri forniti.
- 'ball_tree' : Utilizzerà l'algoritmo BallTree
- 'kd_tree' : Utilizzerà l'algoritmo KDTree
- 'brute' : Utilizzerà una ricerca in forza bruta

Leaf_size: specifica la dimensione delle foglie passata a BallTree o KDTree (default = 30)
P: specifica la metrica di distanza da usare (default = 2).

- ‘1’: distanza di Manhattan
- ‘2’: distanza euclidea

Metric: specifica la metrica di distanza da utilizzare (default = minkwski).

- ‘euclidean’
- ‘manhattan’
- ‘minkowski’

Al fine di selezionare i migliori iperparametri è stato utilizzato il GridSearchCV() e poi si sono calcolati i valori di accuracy, recall, precision e f1

```
Best Parameters K-Nearest Neighbors: {'algorithm': 'auto', 'leaf_size': 10, 'metric': 'euclidean', 'p': 1, 'weights': 'distance'}
                Model  Accuracy  Precision    Recall  F1 Score
0  K-Nearest Neighbors  0.960061   0.938878  0.585462    0.7212
```

Per l’ottimizzazione degli iperparametri del modello Decision Tree sono stati scelti i seguenti valori:

```
param_grid = {
    'criterion' : ['gini', 'entropy','log_loss'],
    'splitter': ['best','random'],
    'max_depth': [10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10, 20],
    'min_samples_leaf': [1, 2, 4, 7]
}
```

Criterion: determina la funzione di misura della qualità delle divisioni fatte nell'albero decisionale (default = ’gini’)

- 'gini': misura l'impurità dei nodi utilizzando l'indice di Gini. Misura l'impurità calcolando la probabilità che un campione estratto casualmente da un nodo sia etichettato in modo scorretto. Un indice di Gini più alto indica una maggiore impurità
- 'entropia': misura l'impurità utilizzando l'entropia, è una misura della casualità o dell'incertezza all'interno di un insieme di dati
- 'log_loss': entropia logaritmica

Splitter: strategia utilizzata per scegliere la suddivisione in ciascun nodo (default = ‘best’)

- ‘best’: per scegliere la migliore suddivisione
- ‘random’: per scegliere la migliore suddivisione casuale.

Max_depth: valore che rappresenta la profondità massima dell'albero (default = 0)

Min_samples_split: Il numero minimo di campioni richiesti per dividere un nodo interno (default = 2)
Min_samples_leaf: Il numero minimo di campioni richiesti per trovarsi in un nodo foglia (default = 1)

Al fine di selezionare i migliori iperparametri è stato utilizzato il GridSearchCV() e poi si sono calcolati i valori di accuracy, recall, precision e f1

```
Best Parameters Decision Tree: {'criterion': 'gini', 'max_depth': 40, 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
          Model  Accuracy  Precision    Recall  F1 Score
0  Decision Tree  0.949487   0.709231  0.724558  0.716812
```

Per l'ottimizzazione degli iperparametri del modello Random Forest sono stati scelti i seguenti valori:

```
param_grid = { 'criterion' : ['gini', 'entropy','log_loss'],
               'n_estimators': [25, 50, 75, 100, 150, 200, 250] }
```

Criterion: determina la funzione di misura della qualità delle divisioni fatte nell'albero decisionale (default = 'gini')
'gini': misura l'impurità dei nodi utilizzando l'indice di Gini. Misura l'impurità calcolando la probabilità che un campione estratto casualmente da un nodo sia etichettato in modo scorretto. Un indice di Gini più alto indica una maggiore impurità
'entropia': misura l'impurità utilizzando l'entropia, è una misura della casualità o dell'incertezza all'interno di un insieme di dati
'log_loss': entropia logaritmica
N_estimators: indica il numero di alberi nella foresta (default = 100)

Al fine di selezionare i migliori iperparametri è stato utilizzato il GridSearchCV() e poi si sono calcolati i valori di accuracy, recall, precision e f1

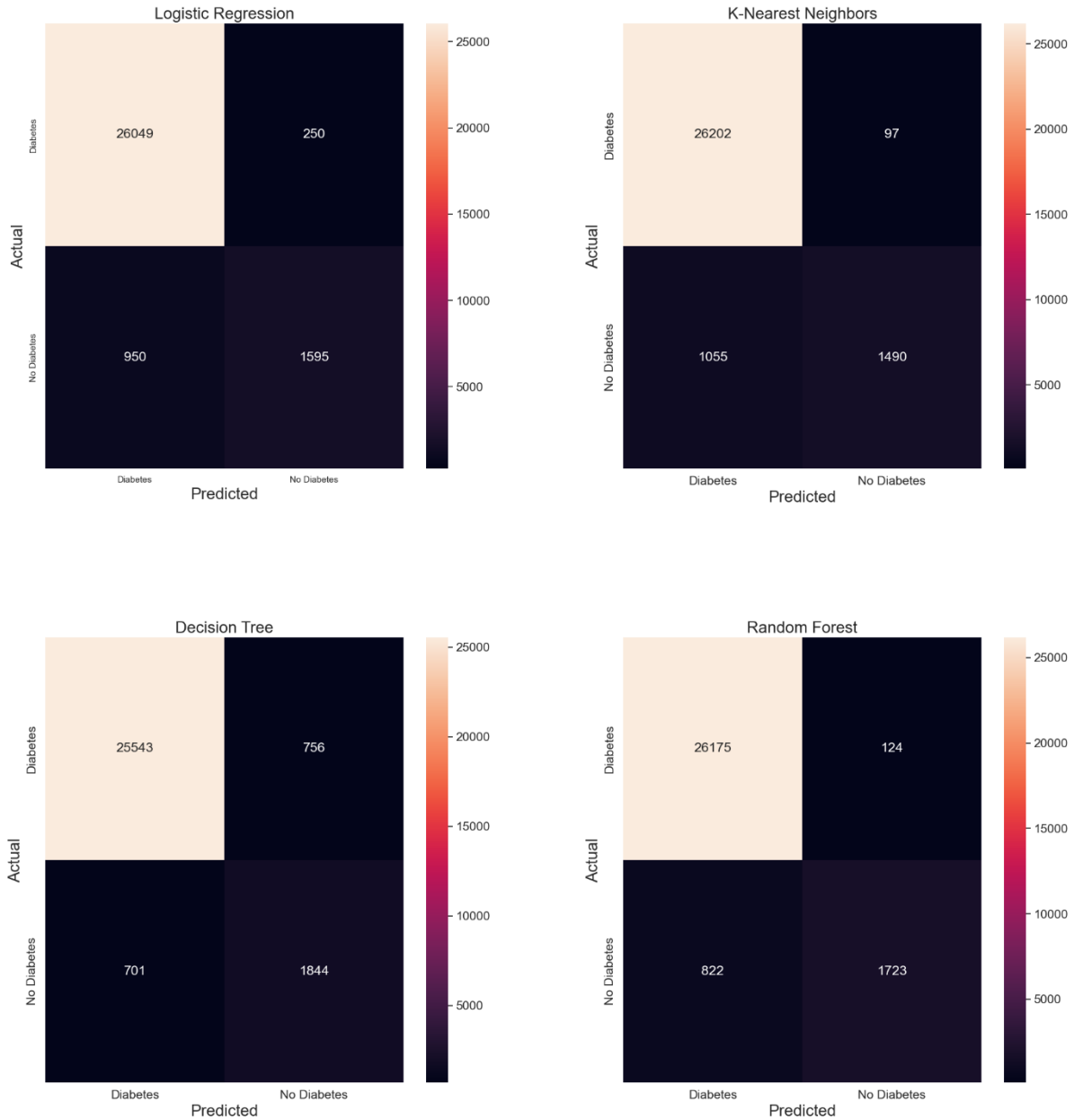
```
Best Parameters Random Forest: {'criterion': 'gini', 'n_estimators': 25}
          Model  Accuracy  Precision    Recall  F1 Score
0  Random Forest  0.677014   0.932864  0.784608  0.967203
```

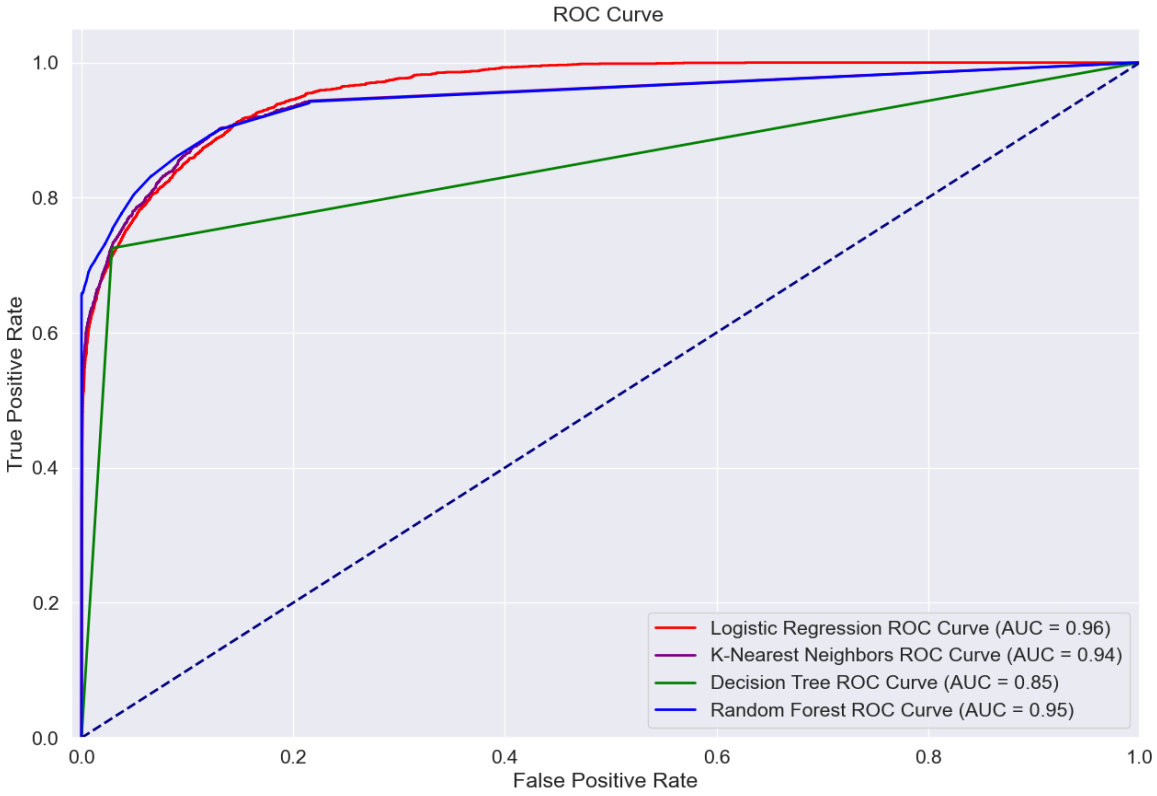
Quindi i risultati ottenuti per tutti i modelli, ordinati secondo il valore di recall , sono i seguenti

```
                Model  Accuracy  Precision    Recall  F1 Score
3       Random Forest  0.677014   0.932864  0.784608  0.967203
2       Decision Tree  0.949487   0.709231  0.724558  0.716812
0  Logistic Regression  0.958397   0.864499  0.626719  0.726651
1  K-Nearest Neighbors  0.960061   0.938878  0.585462  0.721200
```

Dai risultati ottenuti possiamo notare come il Random Forest risulta essere il modello che presenta il valore più alto di recall; quindi, è il modello che ci permette di trovare il più alto numero di casi positivi. Secondo l'obiettivo che ci si è posto, il Random Forest risulta essere il modello migliore.

Per ogni modello sono state visualizzate la matrice di confusione e le curve ROC:





Si può notare come tutti i modelli presentano un comportamento pressoché ideale, si può vedere dal grafico, infatti, che tutti i modelli presentano un valore di AUC piuttosto alto.

BILANCIAMENTO DEL DATASET

APPLICAZIONE DI PESI ALLE CLASSI

Successivamente si è proceduto con il bilanciamento delle classi.

```
#LogisticRegression Balanced
log_reg_bal=LogisticRegression(solver = 'liblinear', class_weight= 'balanced')
grid_log_reg_bal = GridSearchCV(log_reg_bal, param_grid=param_grid_log_reg, cv=skf, scoring='recall').fit(X_train, y_train)
print('\nBest Parameters Logistic Regression Balanced:', grid_log_reg_bal.best_params_)
```

```
#K-nearest Neighbors Balanced
best_k =  bestNeighborsNumber(X_train, y_train,'knnBal')
knn_bal = KNeighborsClassifier(n_neighbors=best_k,  weights='uniform')
grid_knn_bal = GridSearchCV(knn_bal, param_grid=param_grid_knn, cv=skf, scoring='recall').fit(X_train, y_train)
print('\nBest Parameters K-Nearest Neighbors Balanced:', grid_knn_bal.best_params_)
```

```
#DecisionTree Balanced
dt_bal=DecisionTreeClassifier(class_weight='balanced')
grid_dt_bal = GridSearchCV(dt_bal, param_grid=param_grid_dt, cv=skf, scoring='recall').fit(X_train, y_train)
print('\nBest Parameters Decision Tree Balanced:', grid_dt_bal.best_params_)
```

```
#RandomForest Balanced
rf_bal = RandomForestClassifier(class_weight='balanced')
grid_rf_bal = GridSearchCV(rf_bal,param_grid=param_grid_rf, cv=skf, scoring='recall').fit(X_train, y_train)
print('\nBest Parameters Random Forest Balanced:', grid_rf_bal.best_params_)
```

Anche in questo caso al fine di selezionare i migliori iperparametri è stato utilizzato il GridSearchCV() e poi si sono calcolati i valori di accuracy, recall, precision e f1. Si sono ottenuti i seguenti risultati:

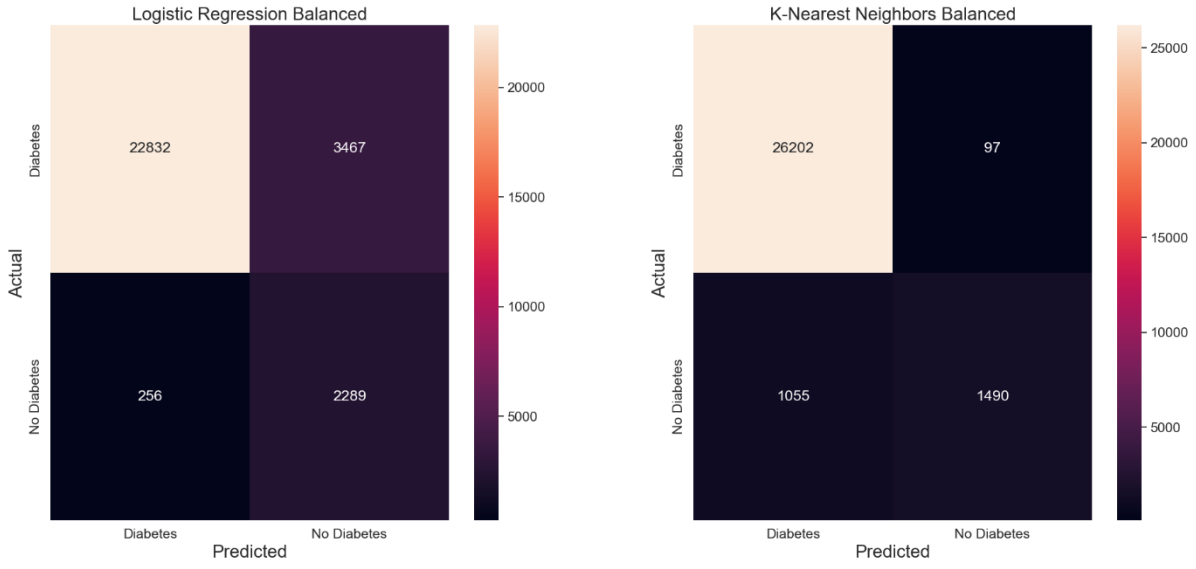
```
Best Parameters Logistic Regression Balanced: {'C': 0.001, 'max_iter': 100, 'penalty': 'l2'}

Logistic Regression Balanced
                       Model  Accuracy  Precision    Recall  F1 Score
0  Logistic Regression Balanced  0.870926   0.397672  0.899411    0.5515


Best Parameters K-Nearest Neighbors Balanced: {'algorithm': 'auto', 'leaf_size': 10, 'metric': 'euclidean', 'p': 1, 'weights': 'distance'}

K-Nearest Neighbors Balanced
                       Model  Accuracy    Recall  Precision  F1 Score
0  K-Nearest Neighbors Balanced  0.960061  0.938878   0.585462    0.7212

Best Parameters Decision Tree Balanced: {'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5, 'splitter':
'random'}

Decision Tree Balanced
                   Model  Accuracy  Precision    Recall  F1 Score
0  Decision Tree Balanced  0.879143   0.415636  0.910806  0.570795

Best Parameters Random Forest Balanced: {'criterion': 'gini', 'n_estimators': 25}

Random Forest Balanced
                   Model  Accuracy  Precision    Recall  F1 Score
0  Random Forest Balanced   0.96703   0.928495  0.678585  0.784109
```

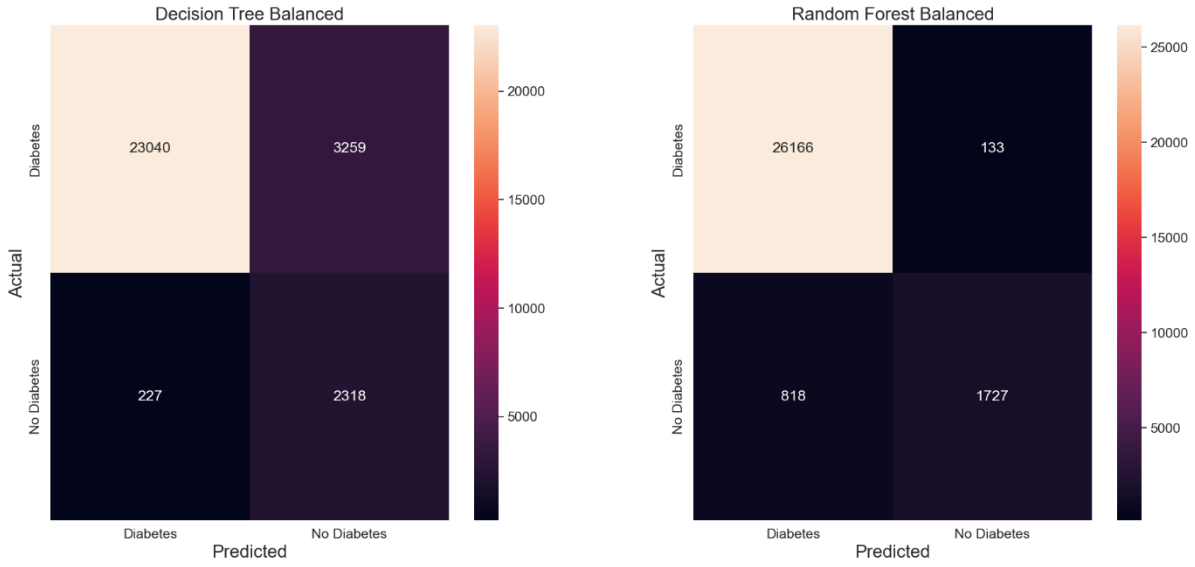
Quindi i risultati ottenuti per tutti i modelli, ordinati secondo il valore di recall, sono i seguenti

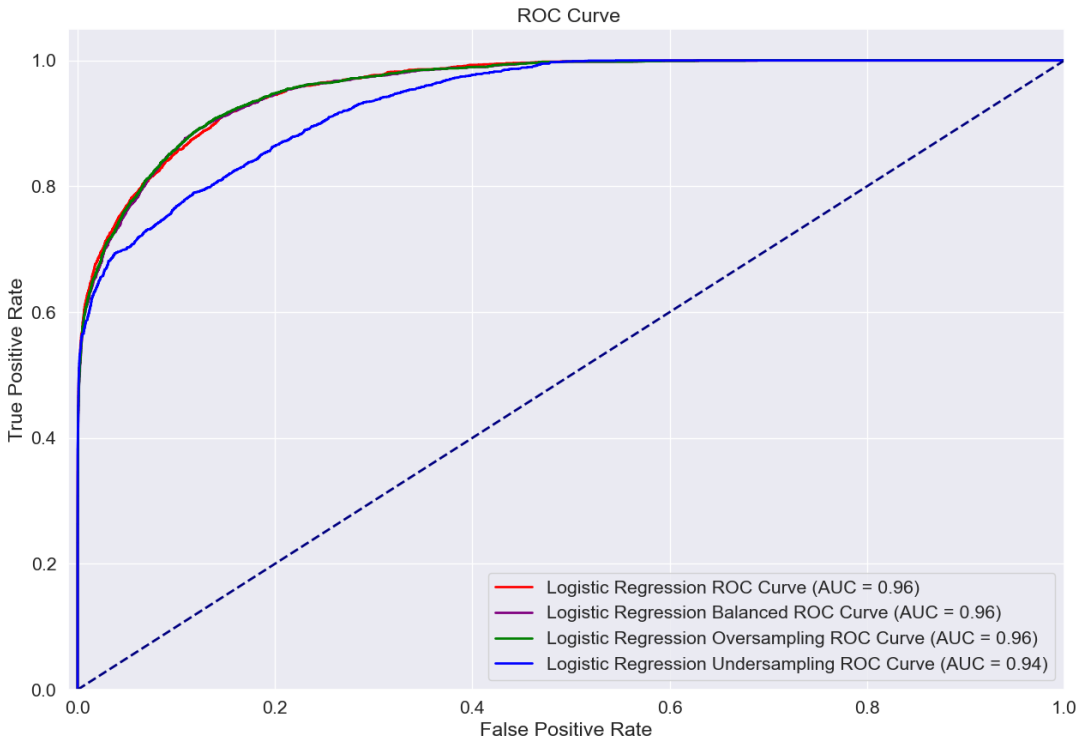
	Model	Accuracy	Precision	Recall	F1 Score
1	K-Nearest Neighbors Balanced	0.960061	0.585462	0.938878	0.721200
2	Decision Tree Balanced	0.879143	0.415636	0.910806	0.570795
0	Logistic Regression Balanced	0.870926	0.397672	0.899411	0.551500
3	Random Forest Balanced	0.967030	0.928495	0.678585	0.784109

Dopo il bilanciamento si può osservare, dai risultati ottenuti, che il modello K-Nearest Neighbors presenta il comportamento migliore con dei valori piuttosto alti per quasi tutte le metriche.

Per ogni modello sono state visualizzate la matrice di confusione e le curve ROC:







Analogamente al caso precedente si può notare che tutti i modelli hanno un AUC alto.

APPLICAZIONE DI RANDOM OVERSAMPLING CON SMOTE

Si è successivamente applicato il random oversampling con SMOTE, si è quindi portato il numero della classe minoritaria (classe 1) alle dimensioni della classe maggioritaria (classe 0).

```
Prima del random oversampling con SMOTE:
Classe 0 no-diabetico: 87664
Classe 1 diabetico: 8482

Dopo il random oversampling con SMOTE:
Classe 0 (no-diabete) dopo SMOTE: 61365
Classe 1 (diabete) dopo SMOTE: 61365
```

Anche in questo caso al fine di selezionare i migliori iperparametri è stato utilizzato il GridSearchCV() e poi si sono calcolati i valori di accuracy, recall, precision e f1. Si sono ottenuti i seguenti risultati:

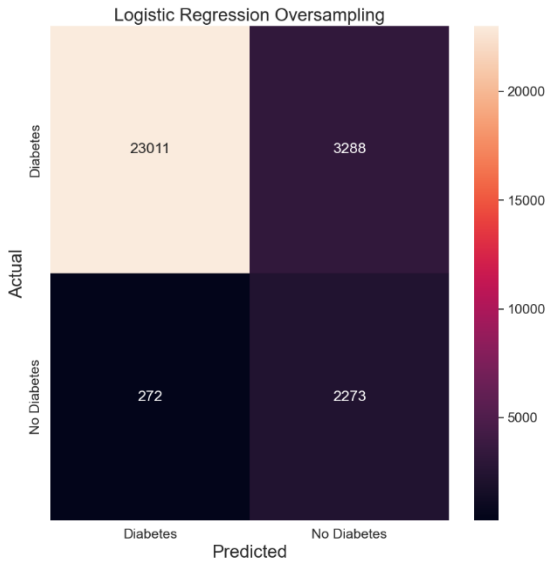
```
Best Parameters Logistic Regression Balanced: {'C': 0.001, 'max_iter': 100, 'penalty': 'l2'}

Logistic Regression Balanced
                      Model  Accuracy  Precision    Recall  F1 Score
0  Logistic Regression Balanced  0.870926   0.397672  0.899411    0.5515


Best Parameters K-Nearest Neighbors Balanced: {'algorithm': 'auto', 'leaf_size': 10, 'metric': 'euclidean', 'p': 1, 'weights': 'distance'}

K-Nearest Neighbors Balanced
                      Model  Accuracy    Recall  Precision  F1 Score
0  K-Nearest Neighbors Balanced  0.960061  0.938878   0.585462    0.7212

Best Parameters Decision Tree Balanced: {'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5, 'splitter':
'random'}

Decision Tree Balanced
                  Model  Accuracy  Precision    Recall  F1 Score
0  Decision Tree Balanced  0.879143   0.415636  0.910806  0.570795

Best Parameters Random Forest Balanced: {'criterion': 'gini', 'n_estimators': 25}

Random Forest Balanced
                  Model  Accuracy  Precision    Recall  F1 Score
0  Random Forest Balanced   0.96703   0.928495  0.678585  0.784109
```

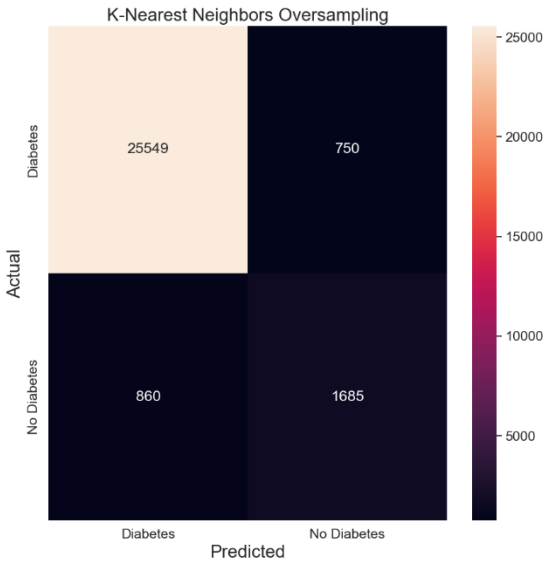
Quindi i risultati ottenuti per tutti i modelli, ordinati secondo il valore di recall sono i seguenti

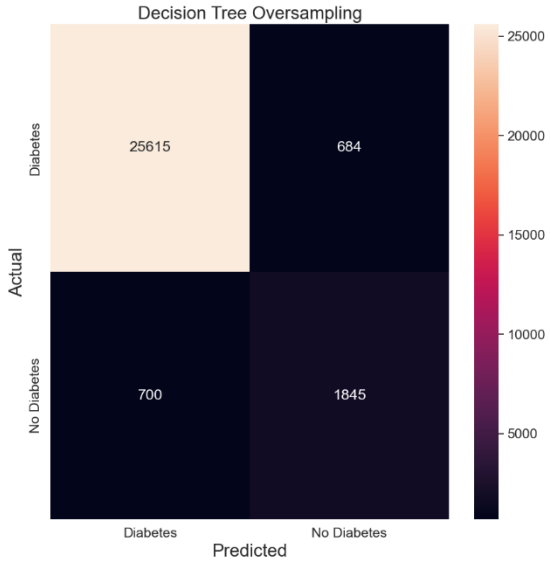
	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression Oversampling	0.876577	0.408739	0.893124	0.560819
2	Decision Tree Oversampling	0.952018	0.729537	0.724951	0.727237
3	Random Forest Oversampling	0.963112	0.853798	0.702161	0.770591
1	K-Nearest Neighbors Oversampling	0.944182	0.662083	0.691992	0.676707

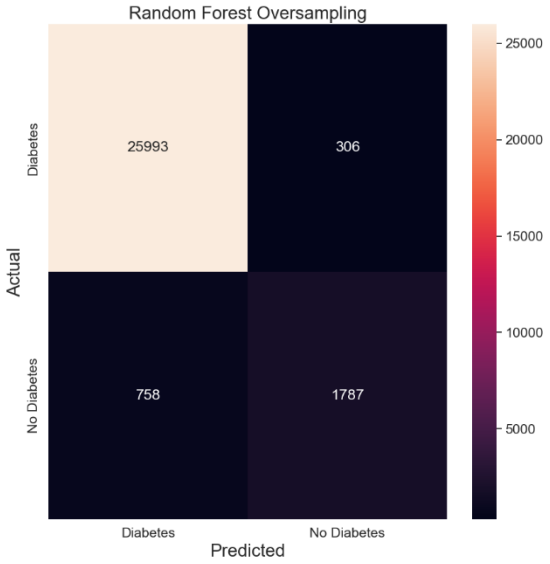
Con l'applicazione del random Oversampling, la Logistic Regression risulta essere il modello con il più alto valore di recall, anche se, a differenza degli altri modelli, presenta dei valori piuttosto bassi per le altre metriche.

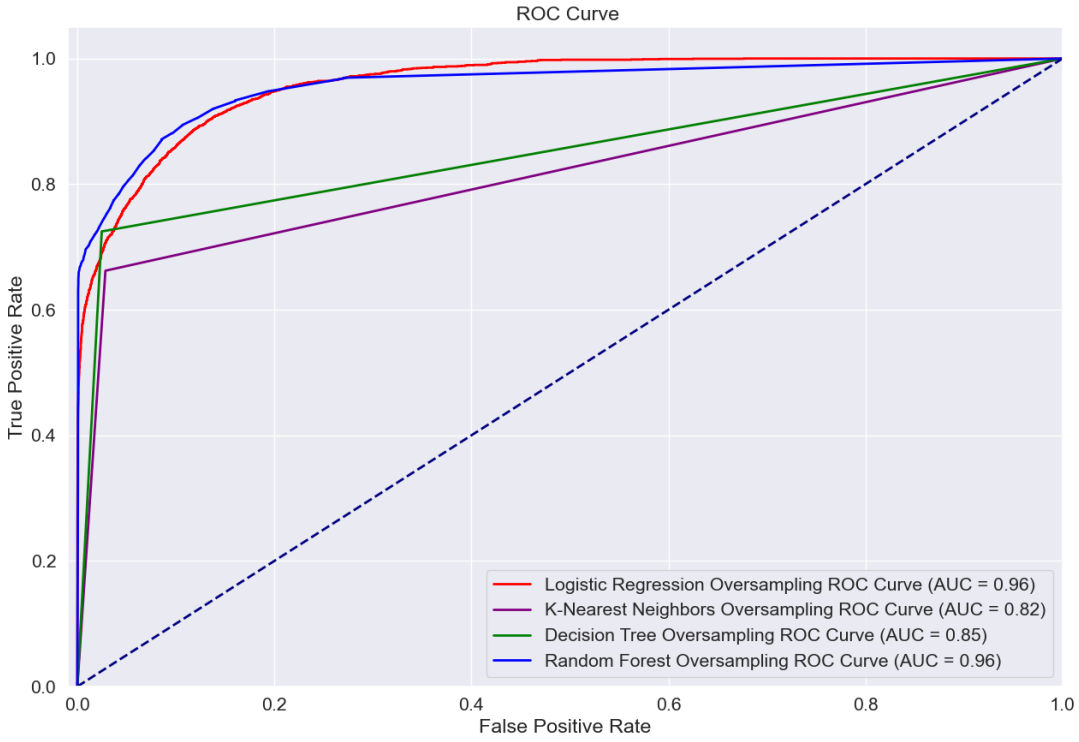
Per ogni modello si è poi visualizzata la matrice di confusione e curva ROC


Logistic Regression Oversampling
Diabetes
No Diabetes
Actual
23011
3288
272
2273
Diabetes
No Diabetes
Predicted
20000
15000
10000
5000


K-Nearest Neighbors Oversampling
Diabetes
No Diabetes
Actual
25549
750
860
1685
Diabetes
No Diabetes
Predicted
25000
20000
15000
10000
5000


Decision Tree Oversampling
Diabetes
No Diabetes
Actual
25615
684
700
1845
Diabetes
No Diabetes
Predicted
25000
20000
15000
10000
5000


Random Forest Oversampling
Diabetes
No Diabetes
Actual
25993
306
758
1787
Diabetes
No Diabetes
Predicted
25000
20000
15000
10000
5000



APPLICAZIONE DI RANDOM UNDERSAMPLING

Infine, si è applicato il random undersampling, si è quindi portato il numero della classe maggioritaria (classe 0) alle dimensioni della classe minoritaria (classe 1).

```
Prima l'undersampling
Classe 0 no-diabetico: 87664
Classe 1 diabetico: 8482

Dopo l'undersampling
Classe 0 no-diabetico: 5937
Classe 1 diabetico: 5937
```

Anche in questo caso al fine di selezionare i migliori iperparametri è stato utilizzato il GridSearchCV() e poi si sono calcolati i valori di accuracy, recall, precision e f1.

Si sono ottenuti i seguenti risultati:

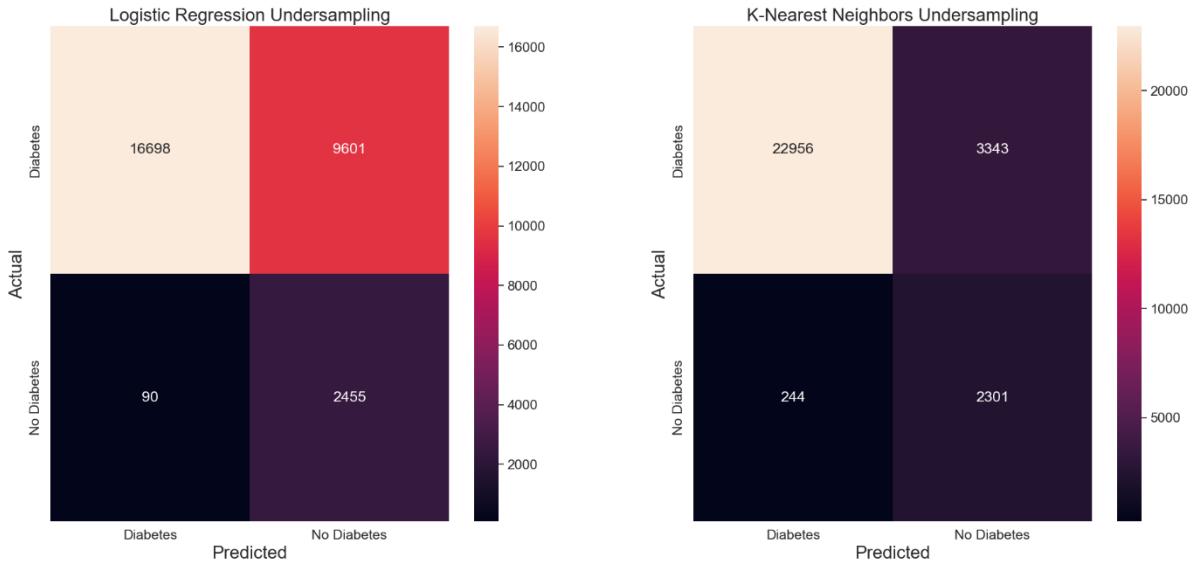
```
Best Parameters Logistic Regression Undersampling: {'C': 0.001, 'max_iter': 100, 'penalty': 'l1'}

Logistic Regression Undersampling
                          Model  Accuracy  Precision    Recall  F1 Score
0  Logistic Regression Undersampling   0.66402   0.203633  0.964637  0.336278


Best Parameters K-Nearest Neighbors Undersampling: {'algorithm': 'auto', 'leaf_size': 20, 'metric': 'euclidean', 'p': 1, 'weights': 'dista
nce'}

K-Nearest Neighbors Undersampling
                          Model  Accuracy   Recall  Precision  F1 Score
0  K-Nearest Neighbors Undersampling  0.875641  0.40769   0.904126  0.561973

Best Parameters Decision Tree Undersampling: {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 5, 'spl
itter': 'random'}

Decision Tree Undersampling
                       Model  Accuracy  Precision    Recall  F1 Score
0  Decision Tree Undersampling  0.841631   0.352185  0.946955  0.513421

Best Parameters Random Forest Undersampling: {'criterion': 'gini', 'n_estimators': 250}

Random Forest Undersampling
                       Model  Accuracy  Precision    Recall  F1 Score
0  Random Forest Undersampling  0.899702   0.464763  0.901768   0.61339
```

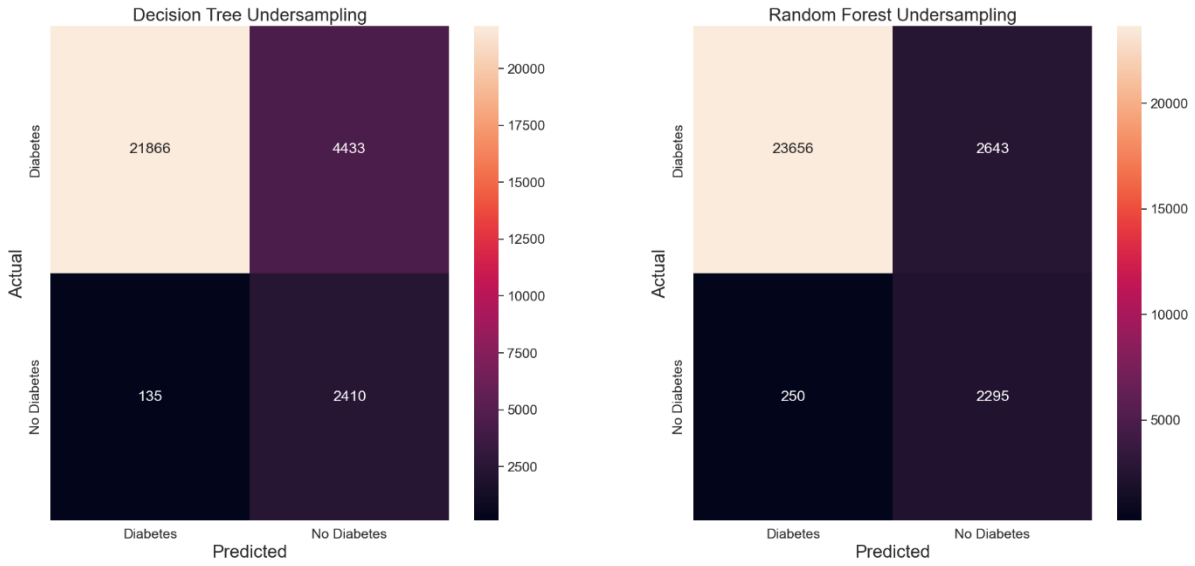
Quindi i risultati ottenuti per tutti i modelli, ordinati secondo il valore di recall, sono i seguenti

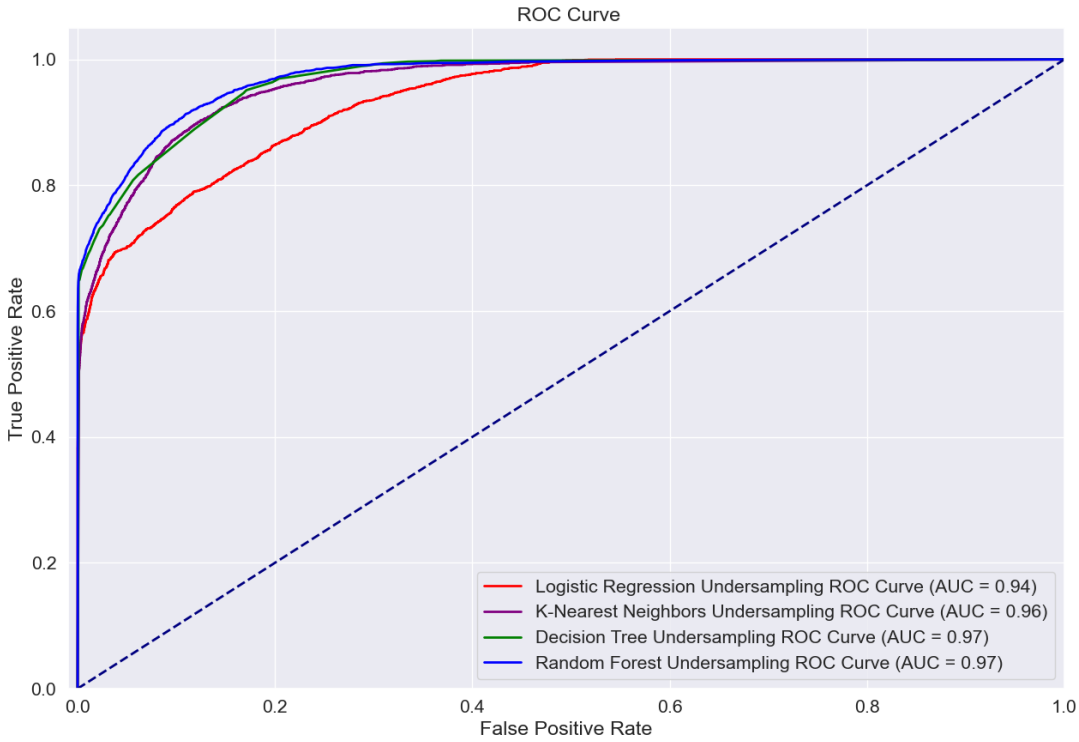
	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression Undersampling	0.664020	0.203633	0.964637	0.336278
2	Decision Tree Undersampling	0.841631	0.352185	0.946955	0.513421
3	Random Forest Undersampling	0.899702	0.464763	0.901768	0.613390
1	K-Nearest Neighbors Undersampling	0.875641	0.904126	0.407690	0.561973

In quasi tutti i modelli è stato ottenuto un valore di recall piuttosto alto a discapito di precision e f1.

Per ogni modello si è poi visualizzata la matrice di confusione e curva ROC







Ogni algoritmo di apprendimento è stato valutato prima e dopo il bilanciamento del dataset con le diverse tecniche

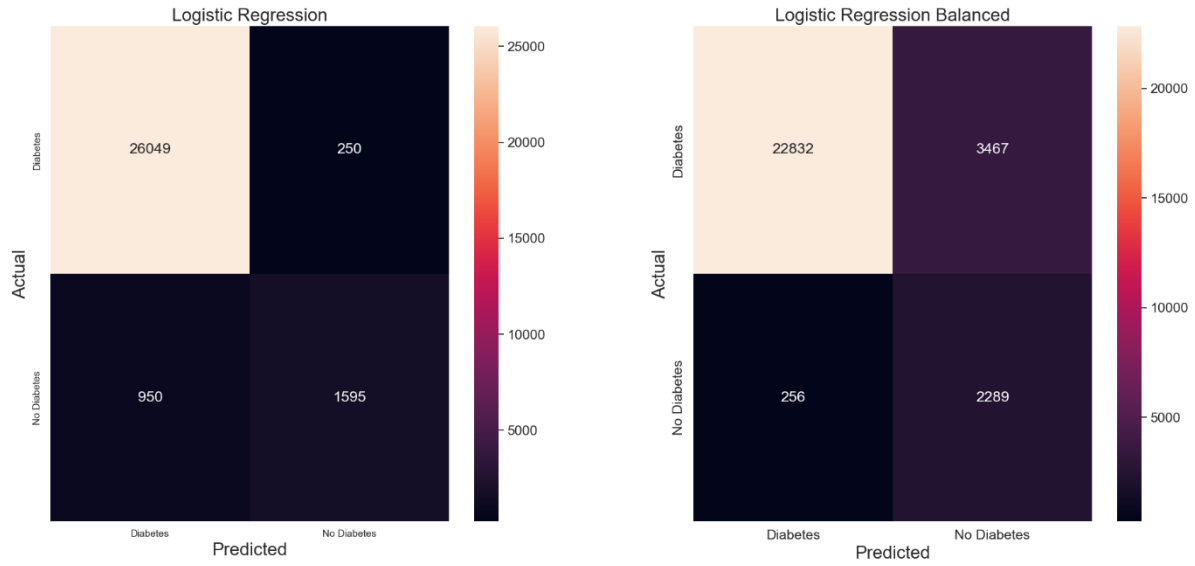
MODELLI A CONFRONTO

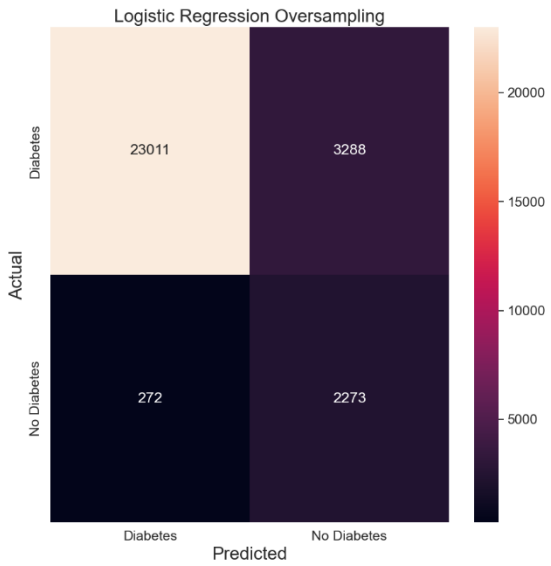
LOGISTIC REGRESSION

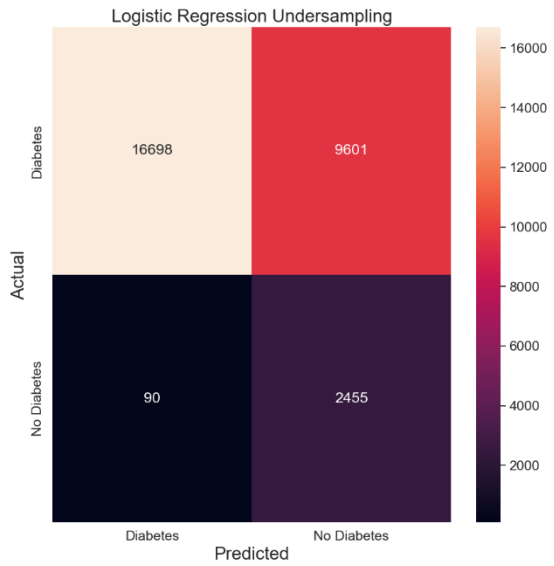
I modelli sono stati ordinati secondo il valore di recall

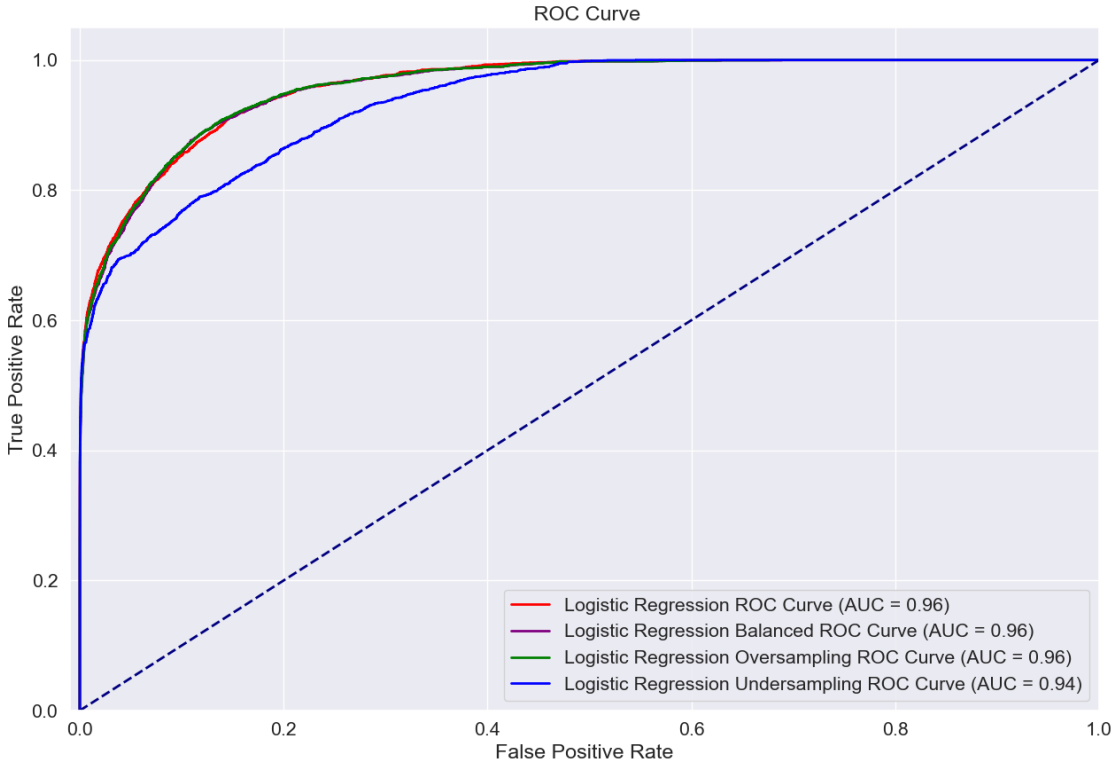
```
Logistic Regression - Logistic Regression Balanced - Logistic Regression Oversampling - Logistic Regression Undersampling

                           Model  Accuracy  Precision    Recall  F1 Score
3  Logistic Regression Undersampling  0.664020   0.203633  0.964637  0.336278
1       Logistic Regression Balanced  0.870926   0.397672  0.899411  0.551500
2   Logistic Regression Oversampling  0.876577   0.408739  0.893124  0.560819
0                Logistic Regression  0.958397   0.864499  0.626719  0.726651
```

Si sono messe a confronto le diverse matrici di confusione e le curve ROC ottenute







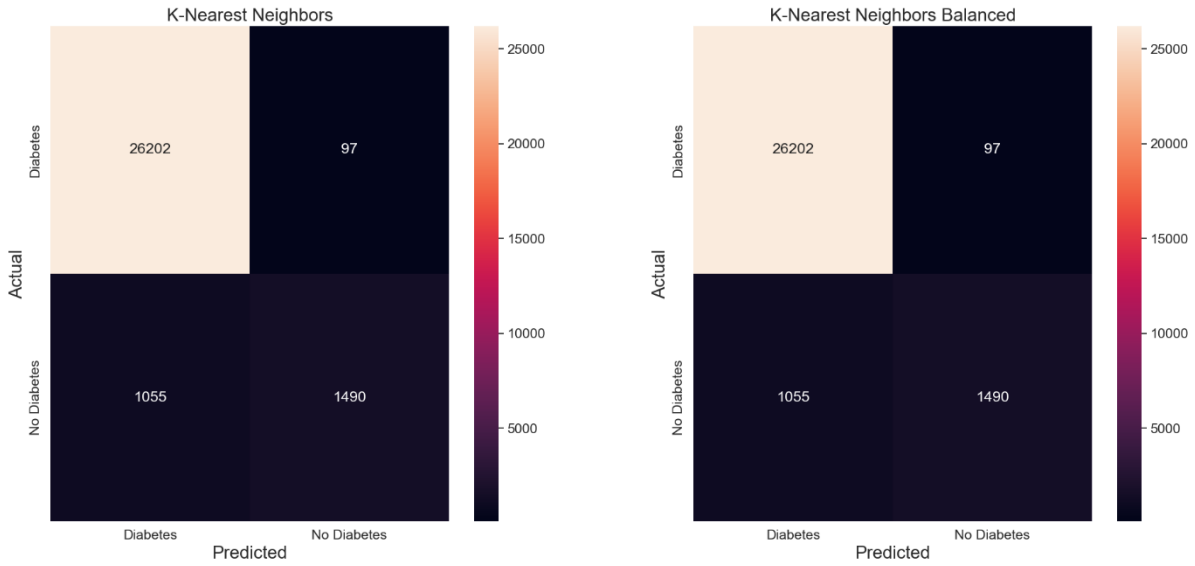


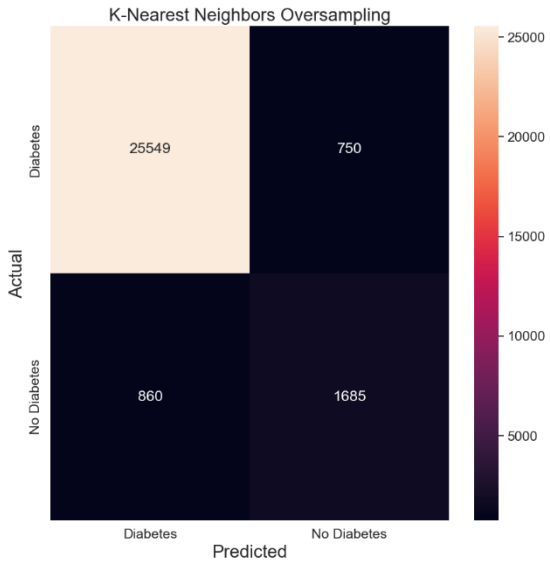
K-NEAREST NEIGHBORS

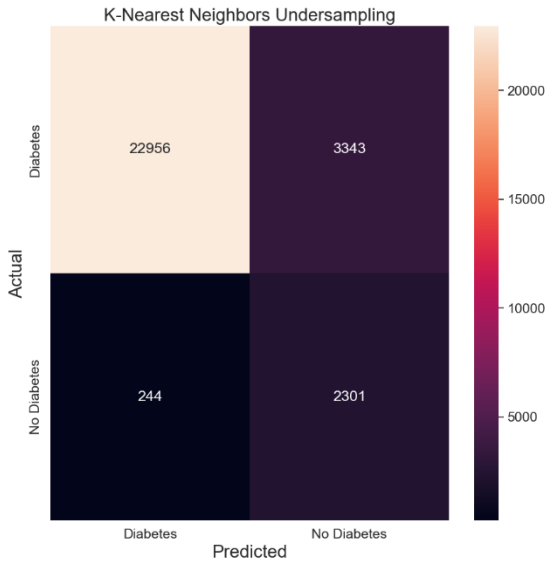
I modelli sono stati ordinati secondo il valore di recall

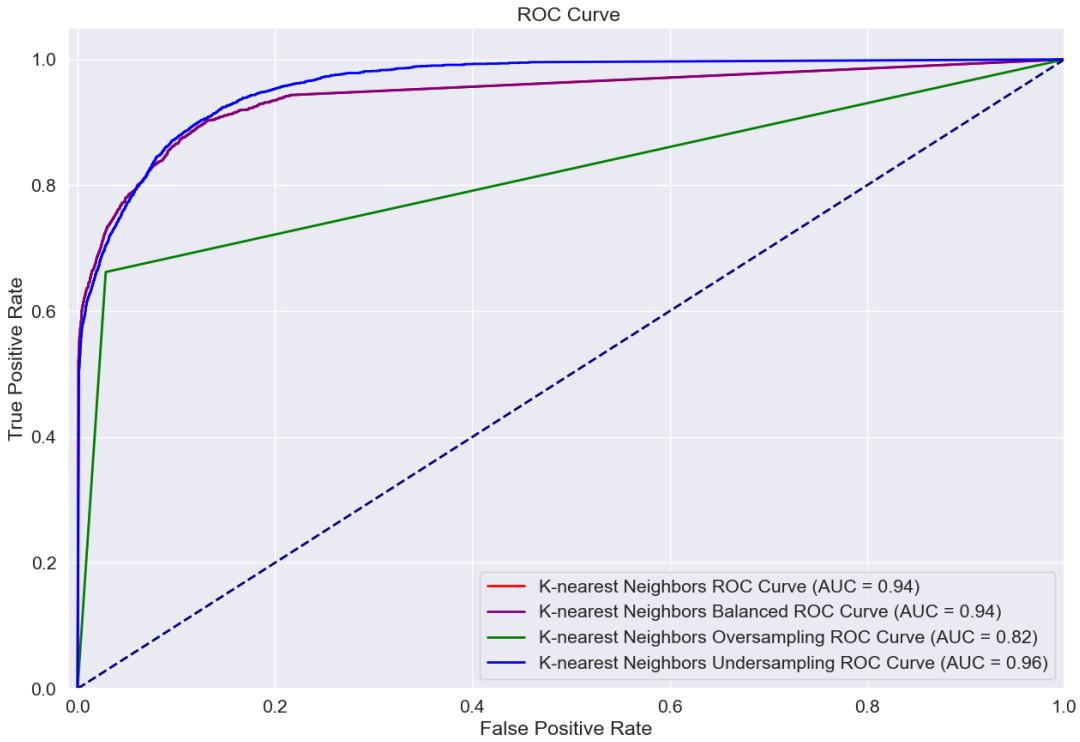
```
K-nearest Neighbors - K-nearest Neighbors Balanced - K-nearest Neighbors Oversampling - K-nearest Neighbors Undersampling

                           Model  Accuracy  Precision    Recall  F1 Score
1       K-Nearest Neighbors Balanced  0.960061   0.585462  0.938878  0.721200
2   K-Nearest Neighbors Oversampling  0.944182   0.662083  0.691992  0.676707
0                K-Nearest Neighbors  0.960061   0.938878  0.585462  0.721200
3  K-Nearest Neighbors Undersampling  0.875641   0.904126  0.407690  0.561973
```

Si sono messe a confronto le diverse matrici di confusione e le curve ROC ottenute




K-Nearest Neighbors Oversampling
Diabetes
No Diabetes
Actual
25549
750
860
1685
Diabetes
No Diabetes
Predicted
25000
20000
15000
10000
5000


K-Nearest Neighbors Undersampling
Diabetes
No Diabetes
Actual
22956
3343
244
2301
Diabetes
No Diabetes
Predicted
20000
15000
10000
5000

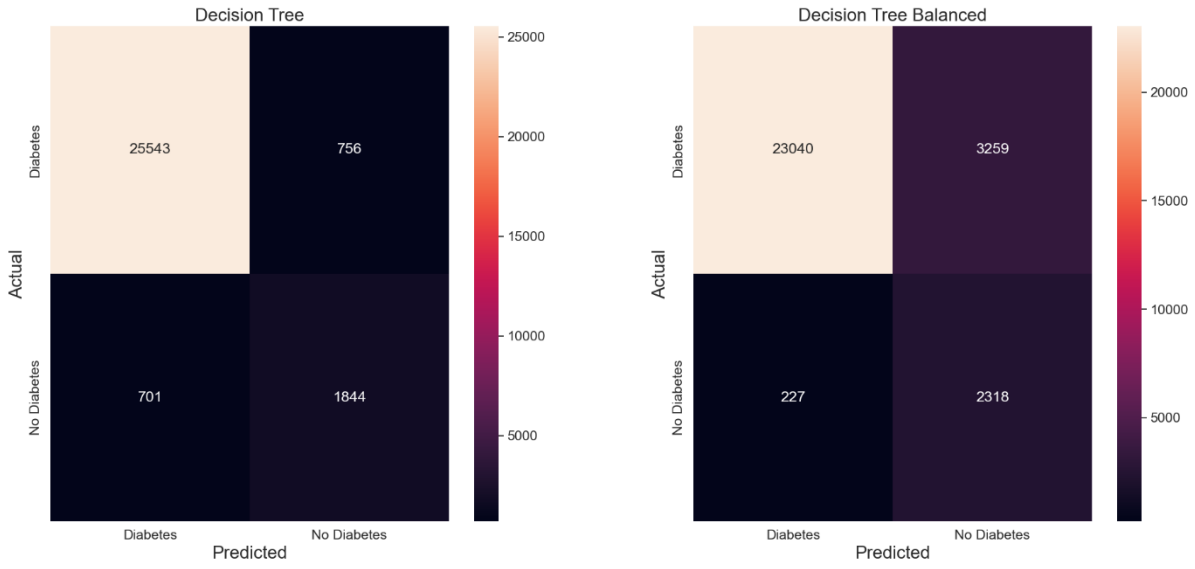

ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
K-nearest Neighbors ROC Curve (AUC = 0.94)
K-nearest Neighbors Balanced ROC Curve (AUC = 0.94)
K-nearest Neighbors Oversampling ROC Curve (AUC = 0.82)
K-nearest Neighbors Undersampling ROC Curve (AUC = 0.96)

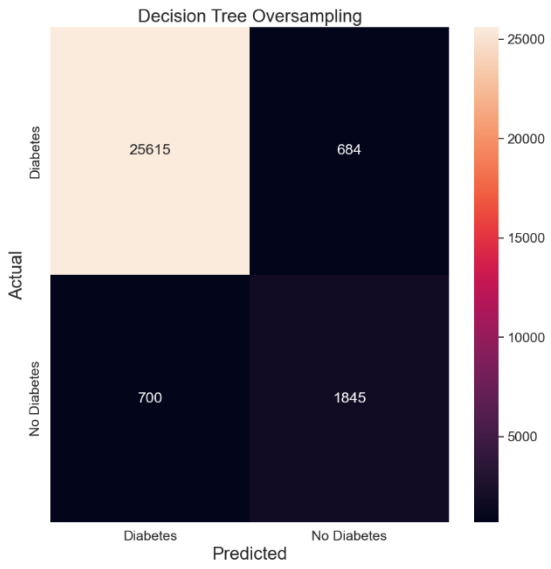
DECISION TREE

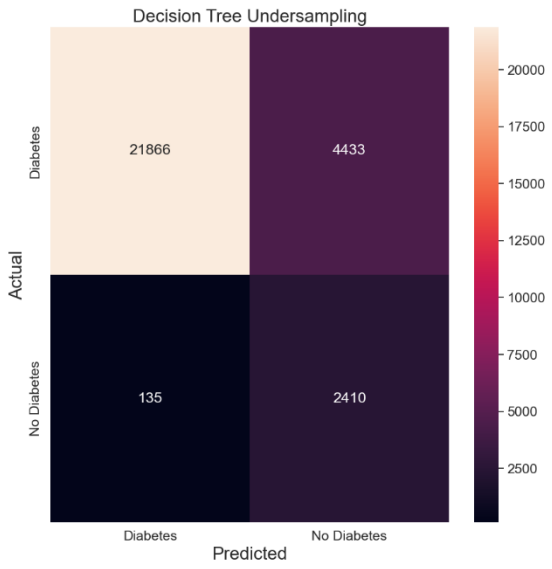
I modelli sono stati ordinati secondo il valore di recall

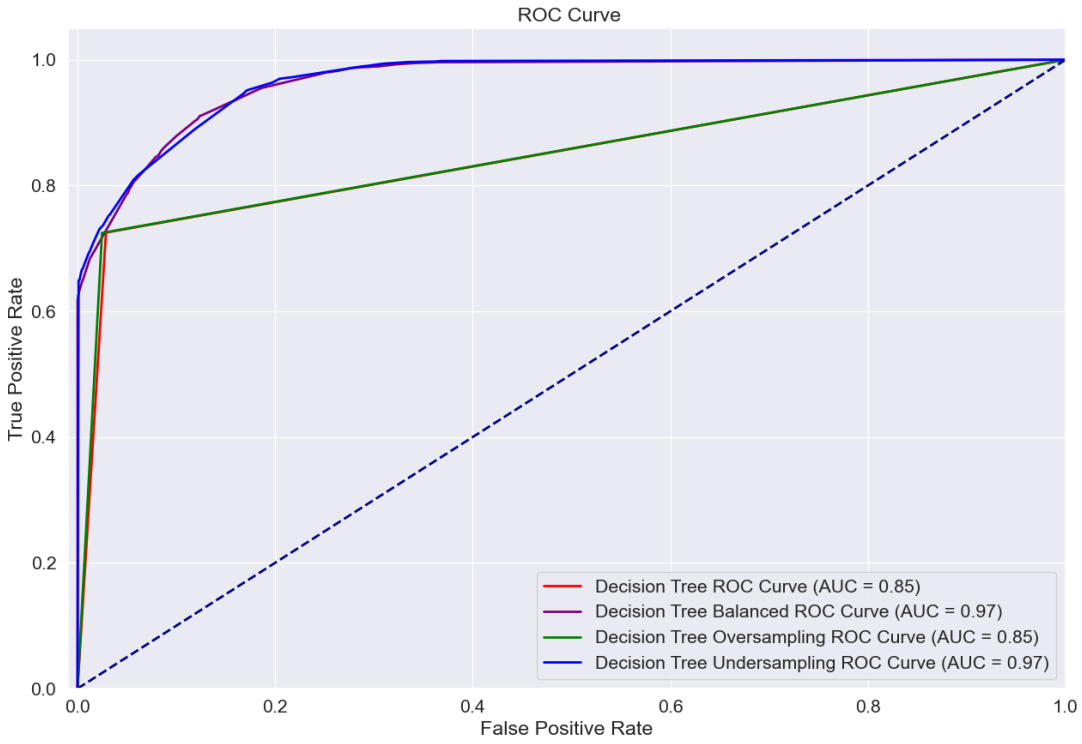
```
Decision Tree - Decision Tree Balanced - Decision Tree Oversampling - Decision Tree Undersampling

                        Model  Accuracy  Precision    Recall  F1 Score
3  Decision Tree Undersampling  0.841631   0.352185  0.946955  0.513421
1       Decision Tree Balanced  0.879143   0.415636  0.910806  0.570795
2   Decision Tree Oversampling  0.952018   0.729537  0.724951  0.727237
0                Decision Tree  0.949487   0.709231  0.724558  0.716812
```

Si sono messe a confronto le diverse matrici di confusione e le curve ROC ottenute







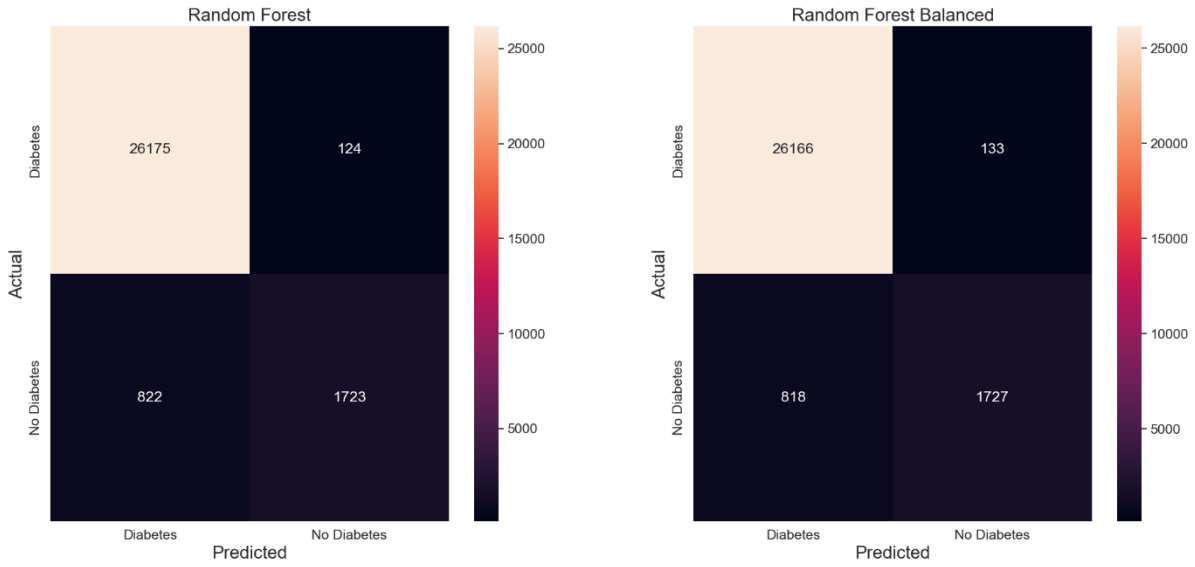


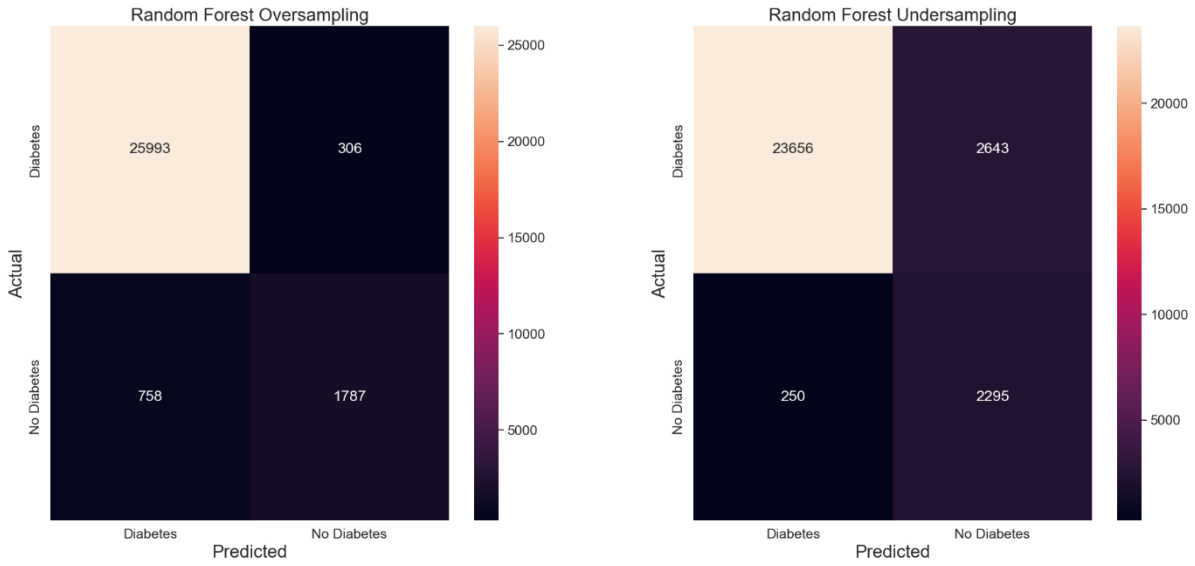
RANDOM FOREST

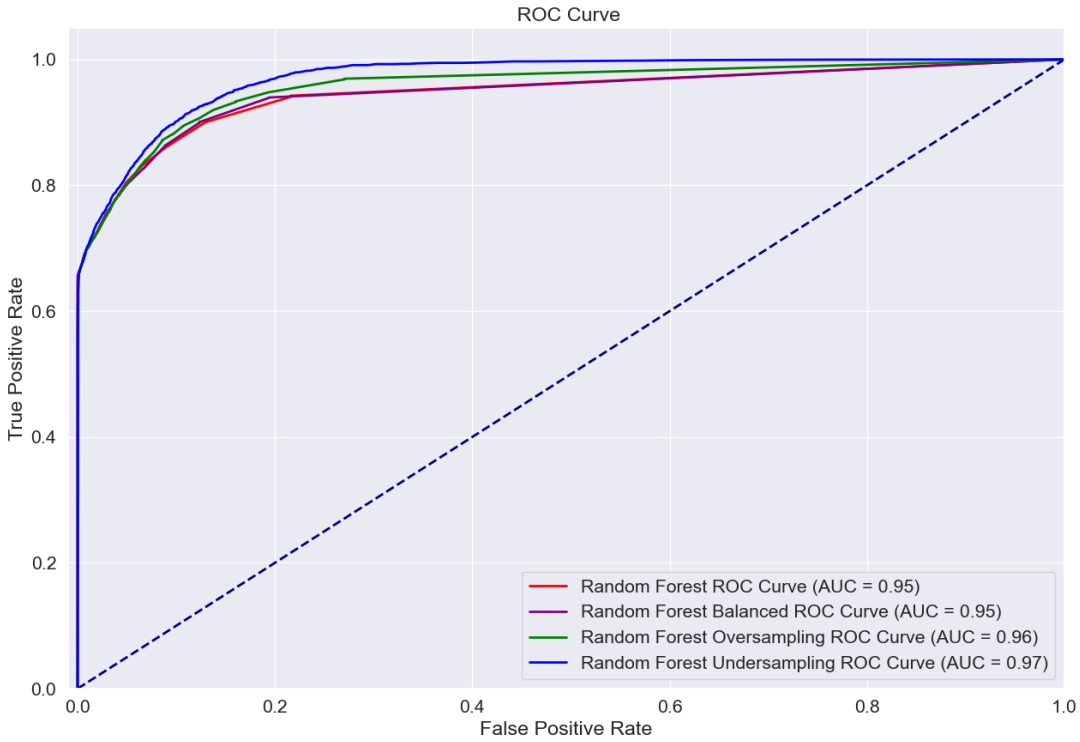
I modelli sono stati ordinati secondo il valore di recall

```
Random Forest - Random Forest Balanced - Random Forest Oversampling - Random Forest Undersampling

                         Model  Accuracy  Precision    Recall  F1 Score
3  Random Forest Undersampling  0.899702   0.464763  0.901768  0.613390
0              Random Forest    0.677014   0.932864  0.784608  0.967203
2   Random Forest Oversampling  0.963112   0.853798  0.702161  0.770591
1       Random Forest Balanced  0.967030   0.928495  0.678585  0.784109
```

Si sono messe a confronto le diverse matrici di confusione e le curve ROC ottenute







Si può concludere per tutti i modelli che si ottiene un valore piuttosto alto di recall quando questi vengono applicati al dataset bilanciato.

APPRENDIMENTO NON SUPERVISIONATO

L'apprendimento non supervisionato è una categoria di apprendimento automatico in cui un modello viene addestrato su dati senza etichette o con etichette parziali.

Nel caso di studio affronteremo il problema di clustering tramite l'algoritmo K-means.

Il clustering consiste nella ricerca di strutture e pattern all'interno di un dataset. Questo approccio è impiegato quando l'obiettivo è individuare gruppi o cluster nei dati senza avere informazioni predefinite o etichette. In altre parole, il clustering è un processo esplorativo che consente di estrarre informazioni significative dal dataset senza la necessità di conoscenze a priori sulle caratteristiche dei dati.

Proprio per la mancanza di etichette sui dati il clustering fa parte degli algoritmi non supervisionati.

L'algoritmo che è stato utilizzato è il K-means, questo inizia con l'assegnazione di k centroidi, k rappresenta il numero di cluster. Per ogni punto nel dataset, viene assegnato il cluster del centroide più vicino. Una volta assegnati tutti i punti ai cluster, i centroidi vengono ricalcolati come la media dei punti nel rispettivo cluster. Questi ultimi due passaggi vengono ripetuti iterativamente fino a quando i centroidi convergono e non ci sono più cambiamenti significativi nelle assegnazioni dei cluster. L'obiettivo dell'algoritmo K-Means è minimizzare la somma dei quadrati delle distanze tra i punti dati e i centroidi dei rispettivi cluster.

Il principale parametro da definire prima di applicare K-Means è il numero K di cluster , in questo caso specifico si è scelto di utilizzare l'indice di Silhouette.

L'indice di Silhouette misura quanto ogni punto dato è simile ai punti del suo stesso cluster rispetto ai punti degli altri cluster.

Il suo valore varia da -1 a 1:

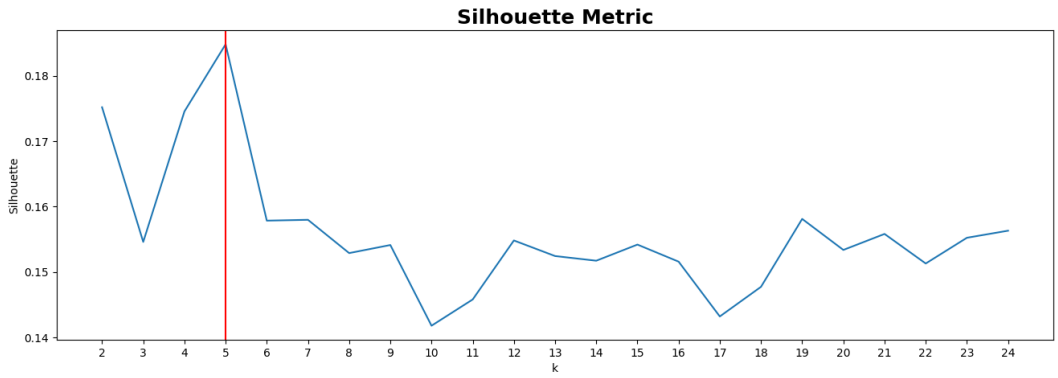
- Un punteggio di 1 indica che i cluster sono ben distanti l'uno dall'altro e chiaramente distinti.
- Un punteggio di 0 indica che i cluster sono indifferenti, ovvero che la distanza tra i cluster non è significativa.
- Un punteggio di -1 indica che i cluster sono assegnati in modo errato.

$$Silhouette\ score\ = \frac{p - q}{max\ (p, q)}$$

Dove p è la distanza media dai punti nel cluster più vicino e q è la distanza media intra-cluster da tutti i punti nel proprio cluster.

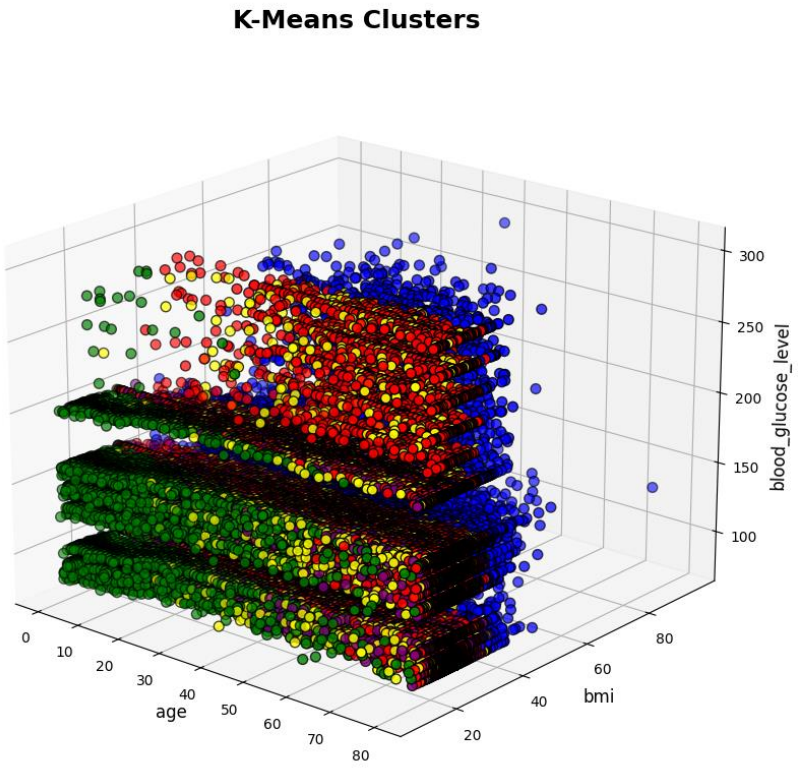
Si è deciso di calcolare l'indice di silhouette su un range di diversi K al fine di trovare il miglior numero di cluster per questo dataset. Si è provato ad iterare K da 2 a 25, per

ognuno di questi si è calcolato l'indice di Silhouette e si è salvato all'interno di un dizionario. I risultati ottenuti sono i seguenti:



Quindi si è scelto di utilizzare k = 5 come numero di cluster.

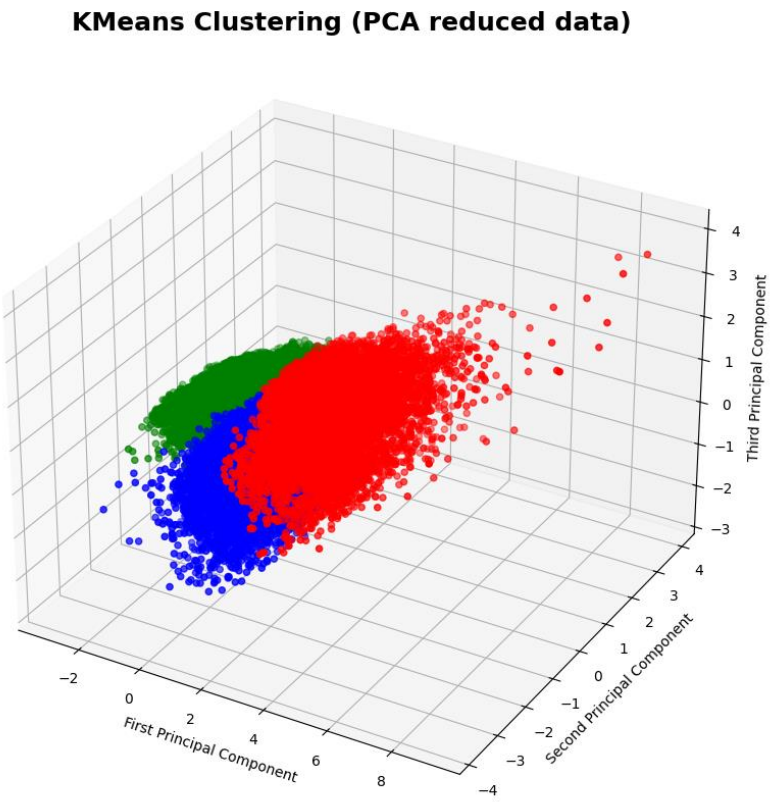
Il grafico ottenuto è il seguente:



Infine, si è deciso di utilizzare la PCA per la riduzione delle dimensioni.

La Principal Component Analysis, o analisi delle componenti principali, è una tecnica utilizzata per semplificare i dati. Date n variabili qualunque è possibile calcolare altrettante componenti principali, operando una combinazione lineare tramite i coefficienti ottenuti dalla matrice degli n auto-vettori della matrice di correlazione delle variabili originali. Le componenti principali così calcolate non sono correlate, hanno varianza pari agli autovalori della matrice di correlazione e conservano tutta la variabilità iniziale dei dati. Inoltre, la prima componente principale (PC1): è la direzione lungo la quale i dati variano di più. Rappresenta la massima varianza nei dati; la seconda componente principale (PC2): è la direzione ortogonale a PC1 e rappresenta la seconda massima varianza nei dati e così via. Ogni successiva componente principale cattura la massima varianza rimanente ed è ortogonale alle precedenti. L'obiettivo della PCA è ridurre la dimensionalità del dataset, mantenendo al contempo la maggior parte possibile della varianza nei dati.

PCA con numero di componenti principali pari a 3



BAYESIAN NETWORK

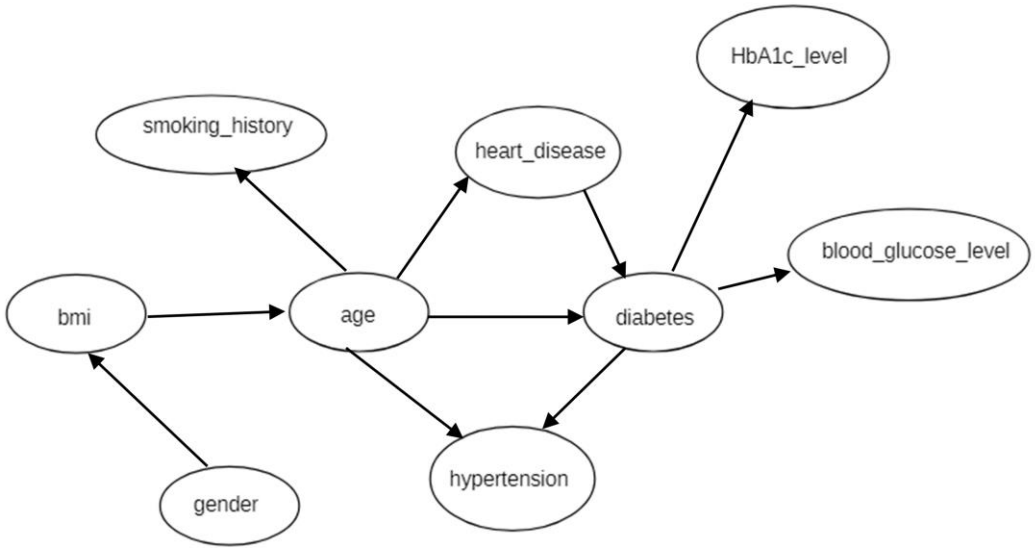
È stata implementata una rete bayesiana per conoscere, in base alle caratteristiche specifiche di un soggetto, la probabilità dell'individuo di sviluppare il diabete.

La rete bayesiana è rappresentata mediante grafi diretti aciclici (DAG). Nel grafo, i nodi rappresentano variabili e gli archi rappresentano relazioni probabilistiche tra di esse.

Nel nostro caso specifico è stata utilizzata la funzione Bayesian Network(), ed è stato creato un oggetto `HillClimbSearch` per condurre una ricerca del miglior modello di rete bayesiana utilizzando il punteggio K2.

La rete bayesiana ottenuta presenta i seguenti nodi e archi :

- Nodi: ['gender', 'age', 'hypertension', 'heart_disease', 'smoking_history', 'bmi', 'HbA1c_level', 'blood_glucose_level', 'diabetes']
- Archi: [('gender', 'bmi'), ('age', 'smoking_history'), ('age', 'diabetes'), ('age', 'hypertension'), ('age', 'heart_disease'), ('heart_disease', 'diabetes'), ('bmi', 'age'), ('diabetes', 'blood_glucose_level'), ('diabetes', 'HbA1c_level'), ('diabetes', 'hypertension')]



Sono state poi utilizzate le capacità di inferenza della rete bayesiana per calcolare le probabilità condizionali.

Sono effettuate due query:

- La prima query calcola la probabilità di un individuo di non avere il diabete (`output`) dati i valori delle variabili di evidenza specificate.

- La seconda query calcola la probabilità di un individuo di avere il diabete (`output`) dati i valori delle variabili di evidenza specificate.

PRIMA QUERY

```
Dati nuovo individuo:
gender: 0,
age: 80.0,
hypertension: 0,
heart_disease: 1,
smoking_history: 1,
bmi: 15.06
HbA1c_level: 6,
blood_glucose_level: 140
```

```
Probabilità per l'individuo di non avere il diabete:
+-------------+-----------------+
| diabetes    |   phi(diabetes) |
+=============+=================+
| diabetes(0) |          0.7489 |
+-------------+-----------------+
| diabetes(1) |          0.2511 |
+-------------+-----------------+
```

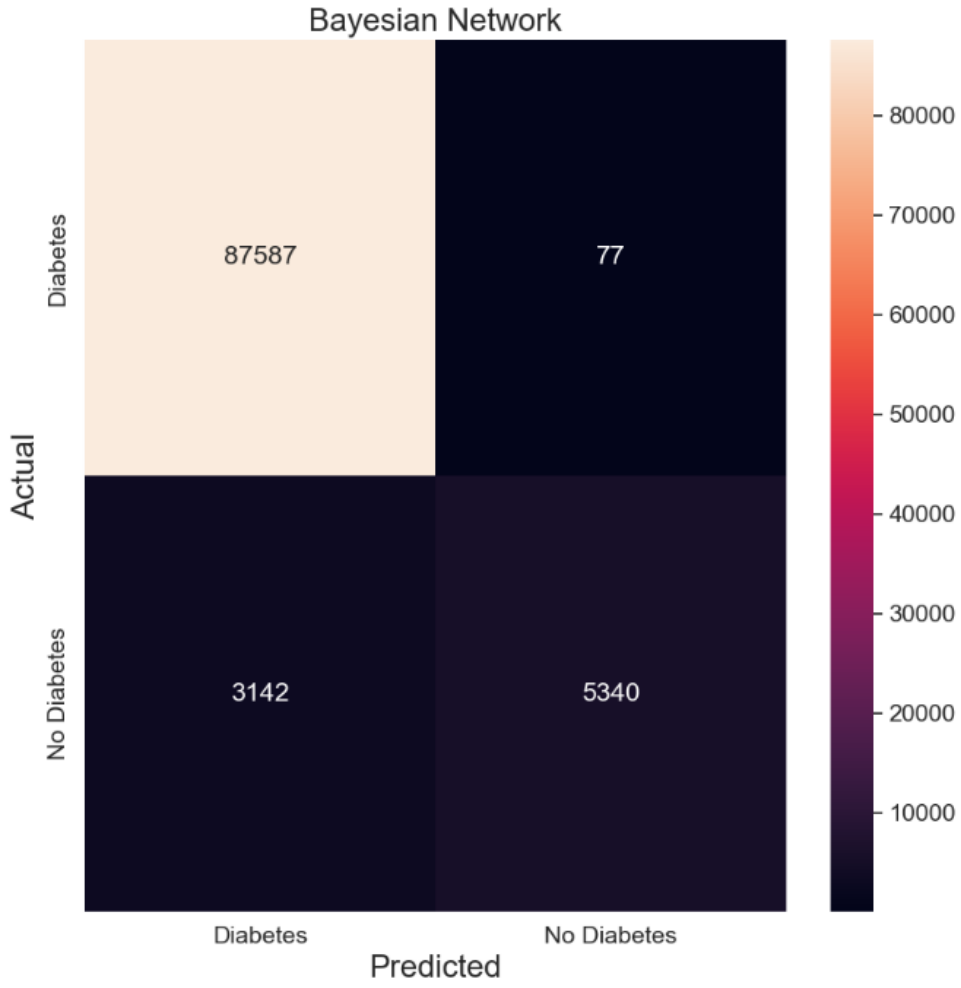
SECONDA QUERY

```
Dati nuovo individuo:
gender: 1,
age: 67.0,
hypertension: 0,
heart_disease: 1,
smoking_history: 4,
bmi:27.32
HbA1c_level: 6,
blood_glucose_level: 200
```

```
Probabilità per l'individuo di avere il diabete:
+-------------+-----------------+
| diabetes    |   phi(diabetes) |
+=============+=================+
| diabetes(0) |          0.5878 |
+-------------+-----------------+
| diabetes(1) |          0.4122 |
+-------------+-----------------+
```

Sono state successivamente valutate le prestazioni della rete bayesiana e questi sono i risultati ottenuti:

```
Classification Report Bayesian Network:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98     87664
           1       0.99      0.63      0.77      8482

    accuracy                           0.97     96146
   macro avg       0.98      0.81      0.88     96146
weighted avg       0.97      0.97      0.96     96146
```


Bayesian Network
87587
77
3142
5340
Diabetes
No Diabetes
Actual
Diabetes
No Diabetes
Predicted
80000
70000
60000
50000
40000
30000
20000
10000