



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica LM-66

RELAZIONE ARTIFICIAL INTELLIGENCE

Evaluation of diabetic metrics

RELATORI

Prof.ssa Loredana Caruccio
Prof.ssa Genoveffa Tortora
Università degli Studi di Salerno

CANDIDATI

Francesco Alfonso Barlotti
Matricola: 0522700022
Salvatore Colucci
Matricola: 0522700024
Domenico Falco
Matricola: 0522700023

Abstract

Il diabete è una delle malattie croniche più diffuse nel mondo e del XXI secolo, rappresentando una delle principali sfide per la salute pubblica. Una diagnosi precoce e accurata può essere fondamentale per prevenire complicanze gravi, migliorando la qualità della vita nei pazienti. Tuttavia, identificare il diabete in fasi iniziali può essere complesso, richiedendo l'analisi di molteplici fattori clinici e diagnostici. Con l'avvento del Machine Learning, è possibile valutare l'efficacia di diversi modelli d'apprendimento automatico nella classificazione dei pazienti in due categorie: "diabetici" e "non diabetici". Le prestazioni dei modelli saranno valutate utilizzando metriche standard quali **accuracy**, **F1-score**, **precision** e **recall**, oltre a strumenti di analisi avanzati come la **confusion matrix** e l'**AUC-ROC curve**. L'obiettivo finale è identificare il modello più performante, confrontando i risultati ottenuti e discutendo le relative implicazioni. Questo approccio permetterà di individuare non solo il miglior modello per il problema in esame, ma anche di comprendere meglio i punti di forza e le limitazioni delle diverse tecniche.

Indice

Elenco delle Figure	iii
1 Introduzione	1
2 Stato dell'Arte	7
3 Obiettivo	10
3.1 Ambito Progettuale	10
3.2 Ambiente di sviluppo e linguaggio di programmazione	11
3.3 Descrizione del workflow	12
3.4 Descrizione del dataset	12
3.5 Suddivisione del dataset	14
3.6 Obiettivo finale	15
4 Metodologia	16
4.1 Analisi del dataset	16
4.2 Rimozione outliers	17
4.3 Feature Engineering	19
4.4 Feature importance	21
5 Implementazione	24
5.1 RandomizedSearchCV	24

5.2	Logistic Regression	29
5.3	Decision Tree	30
5.4	Random Forest	33
6	Valutazione dei risultati	37
6.1	Confusion Matrix	37
6.2	Metriche di valutazione	38
6.3	Curva AUC ROC	39
6.4	Logistic Regression	39
6.5	Decision Tree	41
6.6	Random Forest	42
7	Conclusioni	50
	Bibliografia	51

Elenco delle figure

1.1	Grafico a torta che mostra la percentuale delle donne Pima Indiana diabetiche e non	5
3.1	Importazione della libreria scikit-learn e dei relativi metodi utilizzati.	11
3.2	Importazione delle librerie Pandas e Numpy.	11
3.3	Importazione delle libbreie Searborn e Matplotlib.	12
3.4	Colonne presenti all'interno del dataset.	13
3.5	Suddivisione dei dati in training set e test set	14
4.1	Codice del metodo per la visualizzazione e pulizia del dataset.	17
4.2	Stampa del dataset	18
4.3	Output del metodo isnull().	19
4.4	Grafici dei valori per ogni colonna	20
4.5	L'immagine mostra il codice per la rimozione degli outliers.	21
4.6	Heatmap matrice di correlazione e grafico a barre dei dati prima dell'operazione di feature engineering.	21
4.7	Codice del metodo feature_engineering	22
4.8	Codice del metodo per l'aggiunta della nuova feature.	22

4.9	In questa immagine vengono riportate la Heatmap della matrice di correlazione e il grafico a barre dei dati dopo la fase di feature engineering.	23
4.10	Codice del metodo Feature importance.	23
4.11	Grafico Feature importance del modello Random Forest.	23
5.1	Codice per l'utilizzo di RandomizedSearchCV con il modello Logistic Regression.	26
5.2	Codice per l'utilizzo di RandomizedSearchCV con il modello Decision Tree.	27
5.3	Codice per l'utilizzo di RandomizedSearchCV con il modello Random Forest.	28
5.4	Codice per l'utilizzo del modello Logistic Regression.	30
5.5	Codice per l'utilizzo del modello Decision Tree senza l'utilizzo di RandomizedSearchCV.	32
5.6	Codice per l'utilizzo del modello Decision Tree con l'utilizzo di RandomizedSearchCV.	33
5.7	Decision Tree senza l'utilizzo di RandomizedSearchCV.	34
5.8	Decision Tree con l'utilizzo di RandomizedSearchCV.	35
5.9	Codice per l'utilizzo del modello Random Forest.	36
5.10	Modello Random Forest.	36
6.1	Metriche relative ai dati di training sul modello Logistic Regression.	39
6.2	Confusion Matrix del modello Logistic Regression sui dati di training.	40
6.3	Metriche relative ai dati di test sul modello Logistic Regression. . .	41
6.4	Confusion Matrix del modello Logistic Regression sui dati di test. .	42
6.5	ROC Curve del modello Logistic Regression.	43
6.6	Metriche relative ai dati di training sul modello Decision Tree.	43
6.7	Confusion Matrix del modello Decision Tree sui dati di training. . .	44
6.8	Metriche relative ai dati di test sul modello Decision Tree.	45
6.9	Confusion Matrix del modello Decision Tree sui dati di test.	45
6.10	ROC Curve del modello Decision Tree.	46
6.11	Metriche relative ai dati di training sul modello Random Forest.	46

CAPITOLO 1

Introduzione

Il **diabete** è una delle principali malattie non trasmissibili a livello globale che manifesta un numero di casi più che raddoppiato dal 1980. Viene definita come una malattia cronica che si manifesta quando il pancreas non produce una quantità sufficiente di insulina o quando il corpo non utilizza efficacemente l'insulina prodotta; l'insulina è l'ormone che viene prodotto dal pancreas e che permette al glucosio l'ingresso nelle cellule e il suo conseguente utilizzo come fonte energetica. Quando tale meccanismo è alterato, il glucosio si accumula nel circolo sanguigno e tutto ciò può portare a danni cronici e disfunzioni di vari tessuti come reni, cuore, vasi sanguigni e nervi. Ad oggi, non esiste una cura per il diabete, il quale è una combinazione complessa di **fattori genetici, ambientali e comportamentali**. I fattori più incisivi di tale rischio includono la storia familiare di diabete, etnia, età avanzata, sovrappeso, inattività fisica e una dieta squilibrata; inoltre una diagnosi tardiva e la gestione inadeguata del diabete spesso portano allo sviluppo di complicanze severe, come nefropatia, retinopatia e neuropatia, aggravando il peso sanitario ed economico per gli individui e per le società. Inoltre, l'assenza di diagnosi precoce aumenta il rischio di ulteriori malattie croniche. Il diabete lo si può suddividere in due categorie fondamentali:

- **diabete di tipo 1;** [1]

- **diabete di tipo 2.** [1]

Coloro i quali sono affetti da diabete di tipo 1, nonché denominato diabete mellito insulino dipendente, sono perlopiù giovani di età inferiore ai 30 anni; questo tipo di diabete è una **malattia autoimmune cronica** in cui il sistema immunitario del corpo attacca erroneamente le cellule beta del pancreas, responsabili della produzione di insulina. La conseguenza a tale fenomeno è una riduzione progressiva della produzione di insulina. Nonostante le cause dell'autoimmunità non siano ancora del tutto comprese, si ritiene che una combinazione di fattori genetici e ambientali, come infezioni virali o esposizione a specifici antigeni alimentari nei primi anni di vita, possano scatenare la malattia e da un punto di vista clinico, i sintomi più comuni includono un rapido aumento della sete, una minzione frequente, perdita di peso non intenzionale, stanchezza cronica e visione offuscata. I livelli di glucosio nel sangue sono spesso molto elevati (iperglycemia) già al momento della diagnosi, e i pazienti possono anche manifestare un'acidosi diabetica, una condizione pericolosa causata dall'accumulo di corpi chetonici nel sangue. Poiché il diabete di tipo 1 non può essere gestito in modo efficace con soli farmaci orali, i pazienti richiedono una terapia insulinica quotidiana per sopportare alla carenza di insulina. Questa terapia prevede regolari iniezioni sottocutanee o l'uso di pompe per insulina per mantenere i livelli glicemici entro limiti accettabili.

Il diabete di tipo 2 rappresenta la forma più comune di diabete, rappresentando circa il 90% dei casi a livello globale. Questa patologia si sviluppa in persone di mezza età o anziane, ma negli ultimi anni si è verificato un preoccupante aumento dell'incidenza tra giovani adulti, adolescenti e persino bambini. La principale caratteristica fisiopatologica del diabete di tipo 2 è l'insulino-resistenza corrispondente alla diminuzione della capacità dei tessuti bersaglio (muscoli/fegato) di rispondere efficacemente all'insulina. La patogenesi del diabete di tipo 2 è multifattoriale, includendo fattori genetici, ambientali e comportamentali. Tra i fattori determinanti abbiamo l'**obesità viscerale**, il **rilascio eccessivo di acidi grassi liberi** e **citochine infiammatorie** da parte del tessuto adiposo, nonché una **dieta squilibrata**, ricca di **zuccheri raffinati e grassi saturi**. La diagnosi è spesso tardiva, in quanto la malattia progredisce lentamente e in modo silente. Sebbene le complicanze acute del diabete

di tipo 2, come la chetoacidosi diabetica, siano rare rispetto al diabete di tipo 1, il diabete di tipo 2 si caratterizza per l'elevata incidenza di complicanze croniche. Queste possono interessare diversi organi e tessuti del corpo, comportando gravi rischi per la salute:

- **Retinopatia diabetica:** causata da un danno ai piccoli vasi sanguigni della retina, che può portare a una progressiva perdita della vista. Le persone con diabete hanno un rischio più elevato di sviluppare altre malattie oculari come glaucoma e cataratta, contribuendo al peggioramento della qualità visiva.
- **Nefropatia diabetica:** la riduzione progressiva della capacità di filtrazione renale può evolvere verso l'insufficienza renale cronica, rendendo spesso necessaria la dialisi o un trapianto di rene. La nefropatia diabetica rappresenta una delle principali cause di malattia renale terminale a livello globale.
- **Malattie cardiovascolari:** nei Paesi industrializzati, oltre il 50% dei decessi nei diabetici è attribuito a malattie cardiovascolari, un dato che porta ad egualizzare il rischio cardiovascolare di un paziente diabetico a quello di un individuo che ha già subito un evento cardiovascolare maggiore.
- **Neuropatia diabetica:** questa complicanza colpisce i nervi periferici e autonomici, manifestandosi in circa il 50% dei pazienti con diabete. Può causare perdita di sensibilità, ulcere cutanee e danni agli arti, che nei casi più gravi possono richiedere amputazioni. Inoltre, le neuropatie possono compromettere la funzione di vari organi, inclusi cuore, occhi, stomaco, e rappresentano una delle principali cause di impotenza maschile.
- **Piede diabetico:** il piede diabetico è considerato la principale causa di amputazioni degli arti inferiori di origine non traumatica, con gravi implicazioni sulla qualità della vita dei pazienti.
- **Complicanze in gravidanza:** nelle donne con diabete, la gravidanza può essere associata a un rischio maggiore di complicanze materno-fetali, tra cui malformazioni congenite, macrosomia (peso eccessivo del neonato) e un aumento della mortalità perinatale. Una gestione attenta del diabete durante la gravidanza è essenziale per prevenire complicazioni.

Un'attenta gestione del diabete di tipo 2[2] si basa su un approccio multidisciplinare che unisce dei cambiamenti dello stile di vita, come una dieta equilibrata e l'aumento dell'attività fisica, con trattamenti farmacologici. Le terapie includono farmaci che migliorano l'insulino-sensibilità, promuovono la secrezione di insulina o riducono il riassorbimento renale del glucosio. Nei casi più avanzati, può essere necessaria una terapia insulinica. Nonostante i due tipi di diabete abbiano in comune l'iperglycemia come segno distintivo, le loro eziologie, caratteristiche cliniche, complicazioni e approcci terapeutici differiscono significativamente. Nel diabete di tipo 1, la perdita totale della capacità di produrre insulina richiede una sostituzione completa con insulina esogena e un monitoraggio continuo dei livelli glicemici per prevenire ipoglicemie o chetoacidosi. Al contrario, nel diabete di tipo 2, l'obiettivo è abbassare l'insulino-resistenza e migliorare la secrezione di insulina residua attraverso strategie farmacologiche e non farmacologiche. Un'altra distinzione può essere fatta sull'impatto sociale e psicologico delle due malattie: i pazienti con diabete di tipo 1 spesso convivono con la necessità di un monitoraggio glicemico continuo e di dosaggi precisi di insulina fin dalla giovane età, influenzando il loro stile di vita. I pazienti con diabete di tipo 2, affrontano una condizione fortemente associata allo stile di vita, e molti possono prevenire o ritardare la malattia con modifiche adeguate alla dieta e all'esercizio fisico.

Gli **indiani Pima**, un gruppo di nativi americani che risiede principalmente in Messico e Arizona sono una popolazione di interesse notevole per la ricerca medica e scientifica a causa dell'elevato tasso di incidenza di diabete che è stato osservato tra i loro membri. Questo fenomeno li ha resi un caso studio particolarmente rappresentativo per l'analisi delle dinamiche e dei fattori di rischio associati al diabete, con implicazioni rilevanti per la comprensione della salute globale; di tale popolazione l'aspetto più importante è una combinazione di **fattori genetici, ambientali e comportamentali** che hanno contribuito al diabete di tipo 2. Questi studi hanno evidenziato una forte componente ereditaria, rendendo gli indiani Pima un campione ideale per esplorare il legame tra genetica e malattia metabolica. Tuttavia, questa predisposizione genetica è ulteriormente aggravata da significativi cambiamenti nello stile di vita tradizionale. Storicamente, gli indiani Pima vivevano seguendo uno stile di vita attivo e si nutrivano di una dieta tradizionale composta prevalentemente da

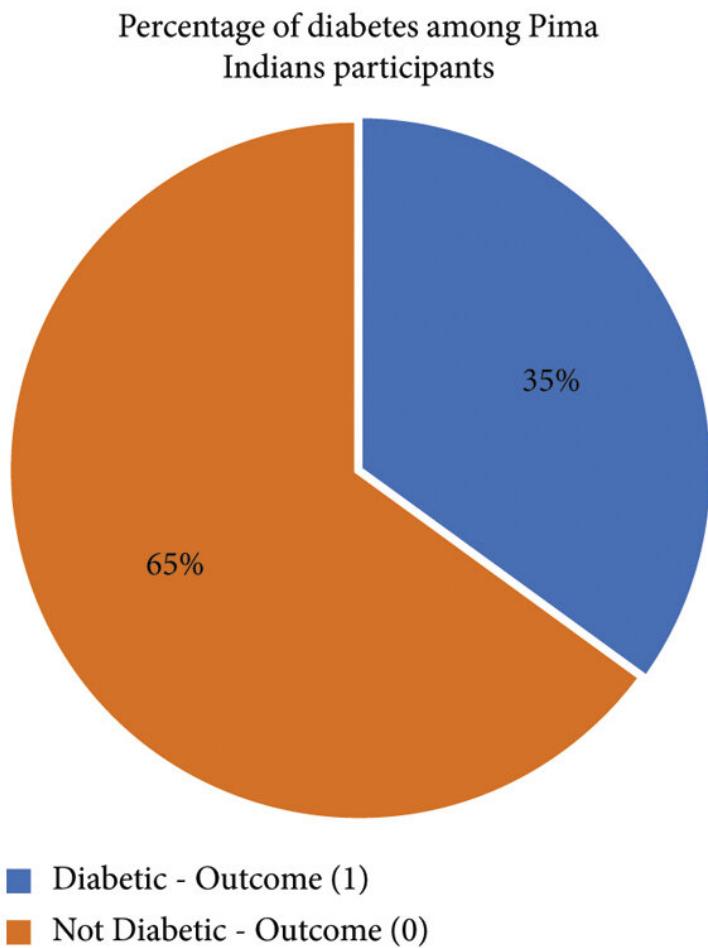


Figura 1.1: Grafico a torta che mostra la percentuale delle donne Pima Indiana diabetiche e non

cibi non raffinati, tra cui mais, fagioli e zucche, che contribuivano a mantenere livelli ottimali di peso corporeo e metabolismo; tuttavia, con l'introduzione di abitudini alimentari tipiche delle società occidentali, caratterizzate da un maggiore consumo di alimenti ricchi di calorie, questa popolazione ha subito un rapido aumento del sovrappeso e dell'obesità, fattori di rischio chiave per lo sviluppo del diabete. Questo scenario fa degli indiani Pima un esempio significativo per l'esplorazione delle complesse interazioni tra **fattori genetici, ambientali e stili di vita** che contribuiscono all'insorgenza del diabete.

Come è possibile evincere dalla figura 1.1 è possibile constatare come la percentuale di persone diabetiche nelle donne di Pima Indiana è superiore rispetto a quelle non diabetiche

Inoltre, le evidenze ottenute dallo studio degli indiani Pima possono essere estrapolate per indagare l'insorgenza e la gestione del diabete in altre popolazioni globali con stili di vita simili o in via di transizione verso diete e attività meno salutari. Data la necessità di affrontare il diabete in modo tempestivo ed efficace, lo sviluppo di strumenti predittivi e diagnostici basati sul machine learning rappresenta una soluzione promettente. Attraverso algoritmi capaci di identificare modelli e correlazioni in set di dati complessi, è possibile individuare i soggetti a rischio, fornire diagnosi accurate e facilitare la gestione del diabete. Questa ricerca si concentra sul confronto tra diverse metodologie di apprendimento automatico, con particolare attenzione agli algoritmi Logistic Regression, Random Forest e Decision Tree, per determinare quale offra un impatto significativo per la classificazione e la previsione del diabete.

CAPITOLO 2

Stato dell'Arte

Il presente capitolo fornisce una panoramica dettagliata dello stato dell'arte nel campo dell'analisi dei dati relativi al diabete, con particolare attenzione ai tre casi di studio che utilizzano il dataset selezionato nel lavoro presentato. Questi casi di studio rappresentano esempi significativi delle diverse metodologie e approcci adottati per l'analisi e la previsione dell'insorgenza del diabete.

Nel primo caso di studio, l'ambito della ricerca scientifica ha approfondito l'utilizzo delle tecniche di apprendimento automatico, in particolare per la diagnosi del diabete. **Nassiwa et al [3]** hanno condotto uno studio comparativo sulla valutazione di diversi modelli, tra cui Logistic Regression (LR), Gradient Boosting (GB) e Random Forest (RF), per prevedere l'insorgenza del diabete. Il dataset utilizzato nello studio è il “Pima Indians Diabetes Database”, suddiviso in un set di training pari all’80% e un set di test pari al 20%. Per garantire l’efficacia del modello, i dati sono stati standardizzati mediante l’uso di StandardScaler(), allo scopo di uniformare le variabili indipendenti che presentavano valori significativamente diversi tra loro. Inoltre, è stato impiegato il metodo **Synthetic Minority Oversampling Technique (SMOTE)** per gestire lo squilibrio del dataset, generando punti dati sintetici per la classe di minoranza. Questa tecnica è stata particolarmente utile, dato che il dataset era di dimensioni ridotte e a bassa dimensionalità. Dai risultati emerge che il modello

Random Forest ha ottenuto le migliori prestazioni tra tutti i modelli analizzati, raggiungendo un'accuratezza dell'81,8%. Utilizzando SMOTE, inoltre, lo squilibrio del dataset è stato migliorato, passando da un bilanciamento del 77% all'82%. Random Forest ha anche fornito importanti indicazioni sui principali predittori dell'insorgenza del diabete, con il glucosio identificato come la variabile più influente, seguito da BMI, età e Diabetes Pedigree Function.

Nel secondo caso di studio, Moon et al. [4] hanno invece condotto un'analisi delle prestazioni di diversi algoritmi per la previsione del diabete, includendo **K-Nearest Neighbors, Logistic Regression, Decision Trees, Random Forest e XGBoost**. Nel loro studio, è stata utilizzata la tecnica **dell'Inter-Quartile Range (IQR)** per rimuovere i valori anomali, ossia dati che si discostavano significativamente dall'intervallo accettabile o che presentavano una volatilità elevata. Per ottimizzare gli iperparametri, gli autori hanno adottato la tecnica dell'ottimizzazione bayesiana, iterando su diversi set di parametri per identificare quelli che offrivano le migliori prestazioni. I modelli sono stati valutati mediante la validazione incrociata k-fold, la curva ROC e altre metriche standard di valutazione. Tra i modelli analizzati, XGBoost si è distinto per l'accuratezza sorprendente del 99%, con un F1-score di 0,80 e un AUC (Area Under the Curve) di 0,85. Lo studio ha anche esaminato **l'applicazione di tecniche di adattamento del dominio**, dimostrando la flessibilità di diversi modelli di apprendimento automatico. I risultati ottenuti evidenziano il potenziale di questi algoritmi nell'adattarsi a contesti differenti, offrendo previsioni non solo legate alla diagnosi del diabete ma anche in altri ambiti applicativi.

Il terzo studio svolto da Kaina Zhao e Zhiping Wang [5] basato sull'applicazione dell'apprendimento automatico per migliorare la diagnosi e la prevenzione del diabete si focalizza sull'obiettivo di sviluppare un modello in grado di accettare precocemente tale malattia utilizzando un approccio di "stacking", che unisce le capacità predittive di tre algoritmi di apprendimento automatico (SVM, Random Forest e XG-BOOST) integrandoli attraverso un meta-classificatore basato sulla regressione logistica. Secondo l'International Diabetes Federation (IDF), il numero totale di persone affette da diabete è previsto a salire fino a 643 milioni entro il 2030 e a 784 milioni entro il 2045. La diffusione ha portato numerosi studiosi a identificare i principali fattori di rischio associati a questa malattia come per esempio: il team

Barbalho hanno evidenziato la correlazione tra HDL-C, trigliceridi, circonferenza della vita e indice di massa corporea, mentre **Shah** hanno scoperto un'associazione tra alti livelli di LDL-C e diabete di tipo 2. I ricercatori, per testare il modello, hanno utilizzato il dataset Pima Indian Diabetes disponibile nel database UCI; tale dataset inoltre contiene 768 campioni, di cui 268 diabetici e 500 non diabetici con 8 caratteristiche di input e 1 caratteristica di output. La parte di pre-elaborazione viene suddivisa nella rimozione di campioni con pochi valori mancanti e nell'imputazione di valori medi per colonne con numerose mancanze. Tra i vari modelli di machine learning studiati nel caso in esame come il Support Vector Machine e il Random Forest vengono indicati l'Extreme Gradient Boosting (XG-Boost) e lo Stacking, dove il modello XG-Boost migliora i risultati costruendo alberi sequenziali che correggono gli errori dei precedenti, mentre, lo Stacking combina più classificatori (SVM, RF, XG-Boost) utilizzando un modello di livello superiore. Per la valutazione dei modelli sono stati considerati i vari valori nella Confusion Matrix e le metriche utilizzate sono: F1-Score, AUC e Recall. Mediante la combinazione di tre classificatori avanzati, il modello Stacking ha mostrato un vantaggio in termini di accuratezza e altre metriche, in quanto, l'accuratezza del 98.67% è supportata da un F1-Score e una AUC entrambi pari a 0.98 e una recall pari a 0.97. Tali risultati determinano la capacità del modello di ridurre significativamente errori di classificazione. In confronto, i modelli singoli, hanno mostrato livelli inferiori come la SVM che ha registrato un'accuratezza del 75.22%, una F1-Score di 0.61; il RF ha risentito di una precisione più bassa del 72.99% e una Recall del 0.55 e infine XG-Boost ha mostrato il migliore equilibrio tra performance e robustezza con un'accuratezza del 77.88%. Il successo dello stacking è dato dalla capacità di unire le forze di ciascun classificatore e compensare le debolezze relative. Infine possiamo concludere quanto questo studio dimostra l'efficacia di un modello di stacking basato su SVM, RF e XG-Boost; grazie a questa combinazione il modello dimostra una capacità di predizione migliore rispetto ai singoli modelli.

CAPITOLO 3

Obiettivo

Nel capitolo "Obiettivo", si illustra l'ambito del progetto, che si concentra sugli sviluppi di modelli di machine learning per la diagnosi e la classificazione del diabete utilizzando il dataset "Diabetes Dataset". In questa sezione, si descrive anche l'ambiente di sviluppo utilizzato, con l'ausilio del linguaggio Python e l'IDE PyCharm, oltre alle librerie essenziali come scikit-learn, Pandas e Numpy. Viene illustrato inoltre come si sia suddiviso il dataset in training e test set per garantire una valutazione accurata del modello. Infine, si delinea l'obiettivo finale del progetto, che mira a creare un modello non solo preciso ma anche robusto e applicabile in contesti reali, evidenziando l'importanza della sua affidabilità oltre la semplice accuratezza.

3.1 Ambito Progettuale

L'aspirazione è la progettazione e la valutazione dei modelli di Machine Learning al fine di determinare la diagnosi e la classificazione del diabete, prendendo in riferimento il dataset "**Diabetes Dataset**", reperibile su Kaggle. Il dataset offre una visione completa di informazioni diagnostiche relative a donne di origine Pima indiana, d'età pari o superiore a ventuno anni d'età.

```
● ● ●  
from sklearn.feature_selection import mutual_info_classif  
from sklearn.model_selection import train_test_split  
from sklearn.preprocessing import LabelEncoder  
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, confusion_matrix, \  
    classification_report, roc_curve, auc  
from sklearn.model_selection import RandomizedSearchCV  
from sklearn.linear_model import LogisticRegression  
from sklearn.tree import DecisionTreeClassifier  
from sklearn import metrics  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.tree import plot_tree
```

Figura 3.1: Importazione della libreria scikit-learn e dei relativi metodi utilizzati.

```
● ● ●  
import matplotlib.pyplot as plt  
import seaborn as sns
```

Figura 3.2: Importazione delle librerie Pandas e Numpy.

3.2 Ambiente di sviluppo e linguaggio di programmazione

Per l'implementazione del progetto si è utilizzato il linguaggio di programmazione di **Python**, supportato da librerie ampiamente utilizzate nel campo del Machine Learning e della gestione dei dati. Si sono utilizzate particolarmente:

- **scikit-learn:** per la creazione, addestramento e valutazione dei modelli di machine learning figura 3.1;
- **Pandas e Numpy:** per la manipolazione e l'analisi del dataset figura 3.2;
- **Searborn e Matplotlib:** per la visualizzazione grafica delle analisi e dei risultati figura 3.3.

L'ambiente di sviluppo integrato (IDE) selezionato è PyCharm, che offre una vasta serie di strumenti dedicati allo sviluppo in Python. Questo IDE semplifica in modo significativo attività come la scrittura, il debug e la gestione del codice.

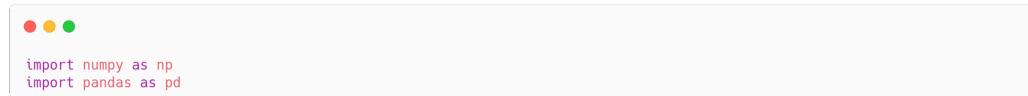


Figura 3.3: Importazione delle librerie Seaborn e Matplotlib.

3.3 Descrizione del workflow

Il workflow è stato suddiviso in diverse fasi principali:

1. **Pre-elaborazione dei dati:** in questa fase si è effettuata un'analisi esplorativa e la rimozione degli outlier. Si è proceduto alla normalizzazione delle variabili numeriche al fine di garantire uniformità tra le scale, questa parte viene discussa meglio nelle sezioni 4.1, 4.2 e 4.3;
2. **Suddivisione del dataset:** il dataset è stato suddiviso in **training set (75%)** e **test set (25%)**. Questa suddivisione è stata progettata per garantire un equilibrio tra l'addestramento del modello e la valutazione delle sue prestazioni su dati non visti, evitando fenomeni di overfitting;
3. **Ottimizzazione degli iperparametri:** per ogni modello sono stati ottimizzati i parametri chiave utilizzando **RandomizedSearchCV**, discusso nella sezione 5.1;
4. **Valutazione del modello:** per ciascun modello sono state misurate le metriche standard come: **Accuracy, Precision, Recall, F1-score e AUC-ROC**. Sono state analizzate le matrici di confusione per comprendere al meglio la distribuzione degli errori, discussi nella sezione 6.

3.4 Descrizione del dataset

Il dataset preso in esame è **Diabetes Dataset**¹ presente su Kaggle. Il contenuto dei dati proviene dal National Institute of Diabetes and Digestive and Kidney Diseases e si propone di prevedere, sulla base di misurazioni diagnostiche, se un paziente sia

¹<https://www.kaggle.com/datasets/mathchi/diabetes-data-set/data>

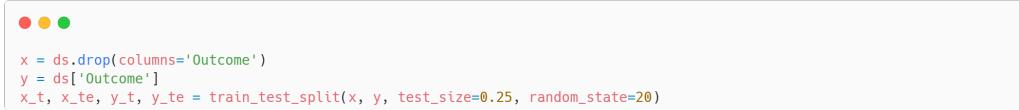
<u>Pregnancies</u>	Numero di gravidanze
<u>Glucose</u>	Concentrazione di glucosio nel plasma
<u>BloodPressure</u>	Pressione sanguigna
<u>SkinThickness</u>	Spessore della piega della pelle del tricipite
<u>Insulin</u>	Livello di insulina nel sangue due ore dopo il test
<u>BMI</u>	Indice di massa corporea
<u>DiabetesPedigreeFunction</u>	Valutazione del rischio familiare di diabete
<u>Age</u>	Età
<u>Outcome</u>	Indicazione della presenza (1) o assenza (0) di diabete

Figura 3.4: Colonne presenti all'interno del dataset.

affetto da diabete. Il dataset è composto esclusivamente da donne di origine Pima indiana, tutte di età pari o superiore a 21 anni. La selezione di queste istanze è stata soggetta a specifici vincoli diagnostici per garantire l'uniformità e la rilevanza delle osservazioni. Il dataset include un totale di 768 istanze e 9 colonne, che comprendono sia caratteristiche indipendenti che la variabile target. Le variabili presenti sono illustrate nella figura 3.4.

Le motivazioni per la scelta del dataset sono state le seguenti:

- **rilevanza della malattia:** il diabete è una malattia cronica che colpisce milioni di persone in tutto il mondo. Il diabete, inoltre, comporta gravi complicanze se non gestito correttamente come malattie cardiovascolari, danni ai reni e perdita della vista. La capacità di predire e classificare accuratamente il diabete può migliorare la gestione preventiva della salute pubblica;
- **caratteristiche del dataset:** il dataset offre l'opportunità di analizzare una popolazione omogenea. Le variabili, presenti nel dataset sono direttamente ed indirettamente collegate, consentendo un'analisi multidimensionale;
- **completezza dataset:** il dataset contiene dati sufficienti, che offrono un'analisi statistica significativa per il training dei modelli di ML. Essendo disponibile su Kaggle, ha ricevuto interesse ed attenzione da altri ricercatori e data scientist, suggerendo di essere stato considerato sufficientemente valido.



```

● ● ●
x = ds.drop(columns='Outcome')
y = ds['Outcome']
x_t, x_te, y_t, y_te = train_test_split(x, y, test_size=0.25, random_state=20)

```

Figura 3.5: Suddivisione dei dati in training set e test set

3.5 Suddivisione del dataset

Il dataset è stato suddiviso in due parti:

- **Training Set (75%):** utilizzato per l’addestramento dei modelli. Questa percentuale garantisce che i modelli abbiano accesso a una quantità sufficiente di dati per apprendere le relazioni tra le variabili indipendenti e la variabile target.
- **Test Set (25%):** utilizzato per la valutazione delle prestazioni dei modelli su dati che non sono stati visti durante l’addestramento. Questo approccio consente di simulare come il modello si comporterà su dati reali e sconosciuti.

La suddivisione 75/25 è stata scelta per garantire una quantità sufficiente di dati per addestrare i modelli, mantenendo al contempo un numero adeguato di dati per testare le loro prestazioni. Questo equilibrio è cruciale per evitare problemi di **overfitting** e/o **underfitting**. Si definisce overfitting quando l’algoritmo si adatta troppo fedelmente o addirittura esattamente ai suoi dati di addestramento, dando luogo a un modello che non è in grado di effettuare previsioni o conclusioni accurate da dati diversi rispetto a quelli di addestramento. L’underfitting, invece, è l’opposto dell’ overfitting: si verifica quando il modello non è in grado di cogliere la relazione tra le variabili di input e di output in modo accurato, generando un elevato tasso di errore sia sul set di addestramento che sia sui dati non visionati.

Nella figura 3.5 viene presentato il codice utilizzato per la suddivisione dei dati, quest’operazione risulta essenziale per garantire una valutazione accurata delle performance del modello, potendo addestrare il modello sui dati di training e poi procedere a testare l’efficacia su un set di dati indipendente. Come primo passaggio, si procede alla preparazione del dataset per la rimozione della variabile target “Outcome” per la creazione del set di features (X). Successivamente, si estrae la

variabile target “Outcome” come vettore delle etichette (y). Utilizzando la funzione `train_test_split` dalla libreria scikit-learn, si è suddiviso il dataset in training set (75%) e test set (25%). Il parametro `test_size=0.25` specifica che il 25% dei dati sarà destinato al test set, mentre `random_state=20` assicura la riproducibilità della suddivisione.

3.6 Obiettivo finale

Il **principale obiettivo** del progetto consiste nello sviluppo di un modello di machine learning in grado di soddisfare le seguenti caratteristiche:

- **Accuratezza:** consentire un’adeguata classificazione tra pazienti affetti da diabete e soggetti non diabetici.
- **Robustezza:** garantire che il modello mantenga performance elevate anche in presenza di scenari imprevedibili o su dati acquisiti dal mondo reale.
- **Rilevanza clinica:** ridurre al minimo i falsi negativi, ovvero i casi in cui un paziente diabetico viene erroneamente classificato come non diabetico, poiché tali errori potrebbero avere gravi implicazioni per la salute.

Le prestazioni dei modelli sono state valutate utilizzando un set di test costituito esclusivamente da dati nuovi, mai impiegati durante le fasi di addestramento. Questo approccio metodologico è stato adottato per ottenere una stima il più possibile realistica circa la capacità del modello di generalizzare sulle applicazioni su dati reali.

CAPITOLO 4

Metodologia

In questa sezione viene indicata la metodologia applicata per effettuare lo studio sul dataset. Il dataset viene presentato nella sezione precedente. Per effettuare lo studio sui dati è stato necessario, per prima cosa, ottimizzare i parametri del dataset, andando ad eliminare eventuali duplicati e valori nulli per garantire risultati ottimali per gli algoritmi di Machine Learning. Le operazioni svolte vengono di seguito riportate:

1. **Analisi del dataset** sezione 4.1;
2. **Rimozione outliers** sezione 4.2;
3. **Feature Engineering** sezione 4.3.

4.1 Analisi del dataset

In questa fase è stato analizzato l'intero dataset. Per avere una panoramica completa dei dati è stato deciso di stampare tutte le righe e colonne del dataset, in modo da esaminare ogni elemento, inoltre sono state estratte informazioni sul numero di righe e colonne, sui tipi di dati associati a ciascuna colonna e sulla quantità di

```

● ● ●
def dataset_analysis(ds):
    # stampa l'intero dataset
    pd.set_option('display.max_rows', None)
    pd.set_option('display.max_columns', None)
    print(ds)
    ds.info() # mostra info sul dataset
    print(ds.isnull().any()) # ricerca di valori nulli
    print(ds.duplicated().sum()) # ricerca di eventuali duplicati

```

Figura 4.1: Codice del metodo per la visualizzazione e pulizia del dataset.

valori non nulli presenti. E' stata, inoltre, identificata la presenza di valori mancanti all'interno del dataset ed è stato effettuato il calcolo del numero di righe duplicate.

Con il metodo in figura 4.1 vengono effettuate le operazioni di "pulizia" del dataset. La libreria Pandas permette di accedere a diversi metodi per la manipolazione dei dati. Con il metodo `set_option()` è possibile impostare un range per la visualizzazione di righe e colonne (nel caso di studio si è scelto di far stampare l'intero dataset). Con le operazioni `isnull()` e `duplicated()` si vanno a ricercare rispettivamente i valori nulli nel dataset, nel caso vengono stampati, e i valori duplicati che vengono sommati e stampati.

Nell'immagine 4.2 viene mostrato per ciascuna colonna il numero di valori non nulli ed inoltre indica anche di che tipo sono i dati presenti all'interno. Nella figura 4.3 viene riportato, per ogni colonna, la presenza di valori non nulli; nel caso di studio sono tutti false in quanto non ci sono valori nulli. In questo caso, infine, la somma dei duplicati dà **0** in quanto nel dataset non sono presenti valori duplicati.

4.2 Rimozione outliers

Prima di passare alla fase di rimozione degli outliers sono stati generati dei grafici per notare come sono strutturati i dati all'interno delle colonne del dataset.

Con i grafici riportati in figura 4.4 si può notare che sono presenti diversi valori che non si trovano all'interno dei range previsti, questi dati, come si può notare, assumono dei valori nulli o superiori alla media. Per evitare di analizzare anche questi dati errati è stata effettuata l'operazione di rimozione degli outliers.

```
Data columns (total 9 columns):
 #   Column            Non-Null Count Dtype  
 --- 
 0   Pregnancies      768 non-null    int64  
 1   Glucose          768 non-null    int64  
 2   BloodPressure    768 non-null    int64  
 3   SkinThickness    768 non-null    int64  
 4   Insulin          768 non-null    int64  
 5   BMI              768 non-null    float64 
 6   DiabetesPedigreeFunction 768 non-null    float64 
 7   Age              768 non-null    int64  
 8   Outcome          768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Figura 4.2: Stampa del dataset

In figura 4.5 viene riportato il codice del metodo per la rimozione degli outliers. Per l'identificazione e rimozione degli outliers si è usato il metodo di **Interquartile Range (IQR)**:

- si calcola prima il primo quartile (Q1) e il terzo quartile (Q3) della colonna;
- l'IQR è la differenza tra Q3 e Q1, avendo stabilito le soglie per gli outliers come 1.5 volte l'IQR al di sotto di Q1 e al di sopra Q3. La funzione restituisce solo le righe che si trovano all'interno di queste soglie, eliminando gli outliers;

Lo **scarto interquartile (IQR)** è la differenza tra il terzo e il primo quartile, ovvero l'ampiezza della fascia di valori che contiene la metà “centrale” dei valori osservati. Lo scarto interquartile è un indice di dispersione, cioè una misura di quanto i valori si allontanino da un valore centrale. L'IQR si calcola sottraendo il primo quartile (Q1) del terzo quartile (Q3):

$$\text{IQR} = Q3 - Q1$$

```

Pregnancies          False
Glucose              False
BloodPressure        False
SkinThickness        False
Insulin              False
BMI                 False
DiabetesPedigreeFunction False
Age                  False
Outcome              False
dtype: bool

```

Figura 4.3: Output del metodo isnan().

- **Q1:** il primo quartile, o 25°percentile, è il valore al di sotto del quale si trova il 25% dei dati.
- **Q3:** il terzo quartile, o 75°percentile, è il valore al di sotto del quale si trova il 75% dei dati.

4.3 Feature Engineering

Prima di passare alla fase di feature engineering sono stati stampati i dati dell'intero dataset.

Nella figura 4.6 viene mostrata la Heatmap (quella a sinistra) della matrice di correlazione dove è possibile vedere tutte le caratteristiche del dataset relative al diabete. I valori di correlazione variano da -1 a 1, dove il valore 1 indica una correlazione positiva perfetta, mentre -1 indica una correlazione negativa perfetta e 0 indica nessuna correlazione. Il grafico a barre (foto di destra) riporta l'informazione mutua tra le caratteristiche e la variabile target. I valori di informazione mutua sono mostrati sull'asse x, indicando la quantità di informazione che ogni caratteristica fornisce riguardo alla variabile target. Un valore più alto di informazione mutua indica che la caratteristica è più informativa per la predizione della variabile target.

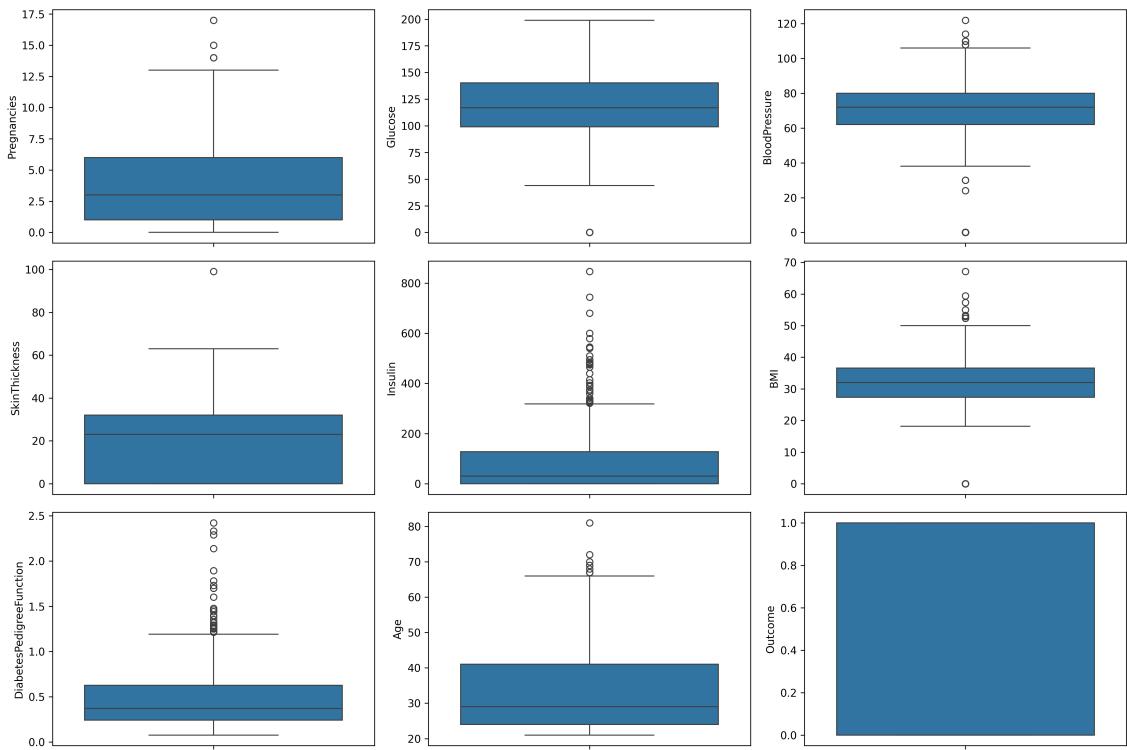


Figura 4.4: Grafici dei valori per ogni colonna

L'operazione di **feature engineering** è stata utile per il processo di trasformazione dei dati grezzi in caratteristiche che meglio rappresentano il problema che stiamo cercando di risolvere, migliorando l'efficacia dei modelli di apprendimento automatico. Il codice del metodo viene riportato nella figura 4.7

Nel codice della figura 4.8, si è provveduto a calcolare l'età minima e massima per comprendere il range dei dati. A seconda dell'età si è categorizzato in gruppi d'età specifici (Young Adults, Middle-Aged Adults, Older Adults, Seniors) mediante dei bin definiti. Quest'aspetto consente di analizzare come l'età influisca su altre variabili dal punto di vista delle categorie. Utilizzando LabelEncoder, si sono convertite queste categorie in valori numerici per facilitare l'analisi e l'uso in modelli di Machine Learning. Si è normalizzata l'età per avere una scala comune da 0 a 1, che potrebbe essere utile per alcuni algoritmi di apprendimento che funzionano meglio con dati normalizzati.

Nella figura 4.9 sono presenti la heatmap della matrice di correlazione e il grafico a barre dei dati dopo l'aggiunta della nuova feature.

```

● ● ●

def remove_outliers(df, column):
    #Calcolo del primo quartile(Q1)
    Q1 = df[column].quantile(0.25)
    # Calcolo del terzo quartile(Q3)
    Q3 = df[column].quantile(0.75)
    # Calcolo dell'intervallo interquartile(IQR)
    IQR = Q3 - Q1
    # Calcolo del limite inferiore
    lower_bound = Q1 - 1.5 * IQR
    # Calcolo del limite superiore
    upper_bound = Q3 + 1.5 * IQR
    # Restituzione dei dati che rientrano nei limiti
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

```

Figura 4.5: L'immagine mostra il codice per la rimozione degli outliers.

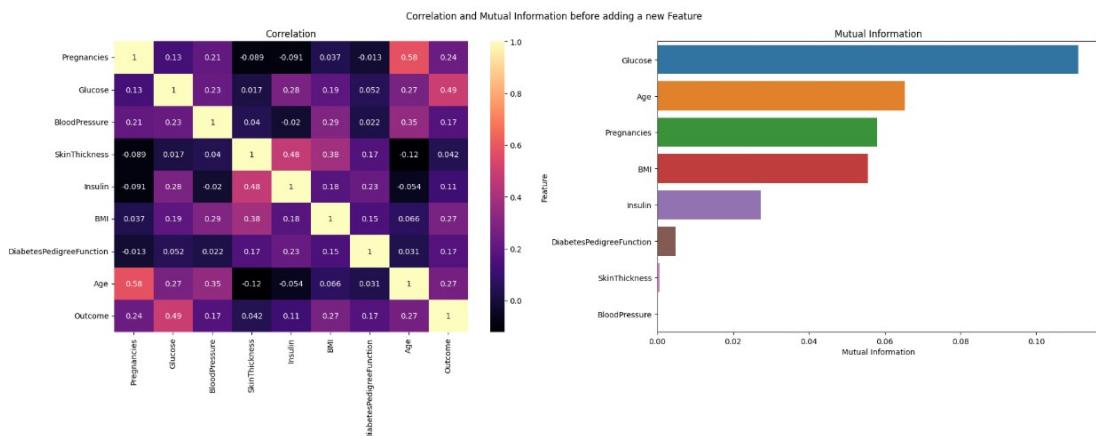


Figura 4.6: Heatmap matrice di correlazione e grafico a barre dei dati prima dell'operazione di feature engineering.

4.4 Feature importance

La valutazione delle Feature importance diventa fondamentale per comprendere le decisioni di modelli quali il Random Forest o Decision Tree siccome determinano una misura quantitativa di quanto ciascuna variabile contribuisca alla previsione. Nel contesto del dataset sul diabete, tali valutazioni ci aiutano a indicare quali fattori sono più significanti nel determinare se un individuo è affetto da diabete o meno. Il grafico di figura 4.11 visualizza le variabili ordinate in ordine decrescente di importanza relativa per il modello, in cui la barra associata ad ogni caratteristica rappresenta la sua influenza sul risultato del modello. Le feature più rilevanti sono:

- **Glucosio**

```

● ● ●

def feature_engineering(df):
    tdf = df.copy() #copia del dataset
    x = tdf.drop(columns='Outcome') #assegno ad x i valori di ogni colonna del dataset tranne quelli della colonna outcome
    y = tdf['Outcome'] #assegno ad y i valori della colonna outcome
    mi = mutual_info_classif(x, y) #calcolo la dipendenza tra x e y
    mi_df = pd.DataFrame({'Feature': x.columns, 'Mutual Information': mi}) #creo di un nuovo dataframe con due colonne
    mi_df = mi_df.sort_values(by='Mutual Information', ascending=False).reset_index(drop=True) #ordino il dataframe

```

Figura 4.7: Codice del metodo feature_engineering

```

● ● ●

def find_min_max(ds):
    #calcola il minimo e il massimo dell'età
    max = ds['Age'].max()
    min = ds['Age'].min()
    print(f"Età minima: {min} Età massima: {max}")

    #definizione intervalli età per la suddivisione
    bins = [20, 36, 51, 66, float('inf')] #split sull'età
    labels = ['Young Adults', 'Middle-Aged Adults', 'Older Adults', 'Seniors'] #definizione delle etichette in base all'età
    ds['Age_Category'] = pd.cut(ds['Age'], bins=bins, labels=labels, right=True) #si assegnano le etichette all'età
    le = LabelEncoder() #conversione delle età in numeri
    ds['Age_Category'] = le.fit_transform(ds['Age_Category'])
    ds['Normalized_Age'] = (ds['Age'] - ds['Age'].min()) / (ds['Age'].max() - ds['Age'].min()) #viene creata una nuova colonna con le età normalizzate

```

Figura 4.8: Codice del metodo per l'aggiunta della nuova feature.

- **BMI**

- **Età**

Mentre le feature meno rilevanti sono:

- **Age_Category**
- **Pressione Sanguigna**

Attraverso tale metodologia possiamo evincere che caratteristiche come Glucosio, BMI, Età sono molto importanti nell'aiutare a classificare i dati poichè tale affermazione ci indica che sia livelli di zucchero nel sangue sono un predittore primario di diabete e sia fattori legati allo stile di vita e al metabolismo. A differenza di variabili come la pressione sanguigna le quali mostrano che il loro contributo complessivo alla classificazione è inferiore.

Nella figura 4.10 viene riportato il codice per l'operazione di feature importance.

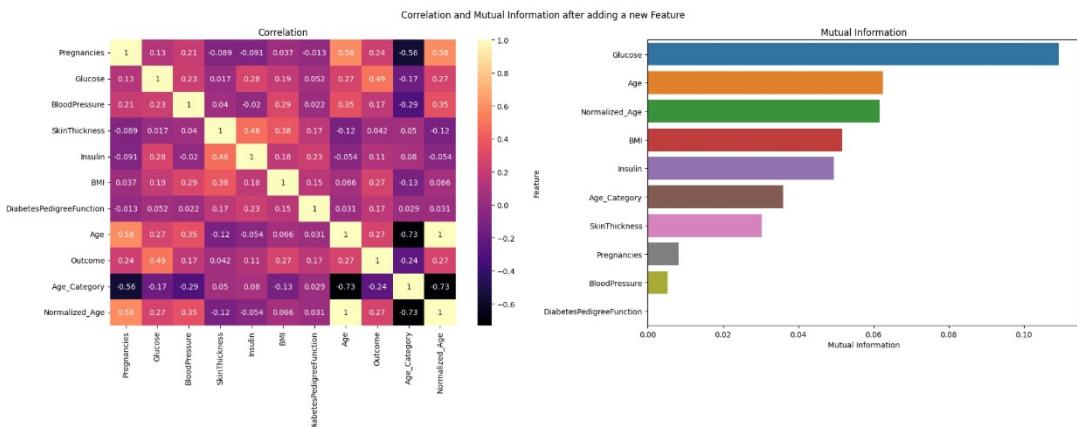


Figura 4.9: In questa immagine vengono riportate la Heatmap della matrice di correlazione e il grafico a barre dei dati dopo la fase di feature engineering.

```

● ● ●

def plot_feature_importance(self, model):
    importances = model.feature_importances_ # Ottiene le importanze delle feature
    indices = np.argsort(importances)[::-1] # Ordina le feature in ordine decrescente di importanza
    features = self.X_train.columns # Ottiene i nomi delle feature
    
```

Figura 4.10: Codice del metodo Feature importance.

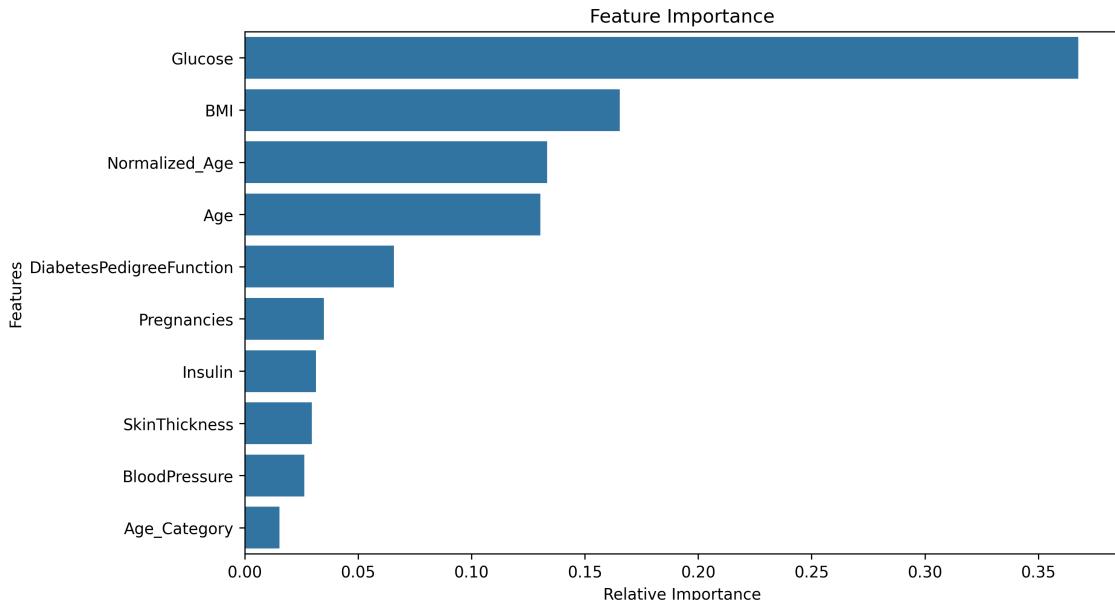


Figura 4.11: Grafico Feature importance del modello Random Forest.

CAPITOLO 5

Implementazione

In questa sezione viene presentata l'implementazione degli algoritmi di Machine Learning utilizzati per il raggiungimento dell'obiettivo finale.

5.1 RandomizedSearchCV

RandomizedSearchCV è una tecnica di ottimizzazione iperparametrica fornita da **scikit-learn**, progettata per individuare i migliori parametri per un modello in modo efficiente. A differenza di **GridSearchCV**, che esplora esaustivamente tutte le possibili combinazioni di parametri all'interno di uno spazio predefinito, **RandomizedSearchCV** seleziona un numero limitato di combinazioni casuali dallo spazio dei parametri specificato. Questa caratteristica rende **RandomizedSearchCV** particolarmente utile quando lo spazio degli iperparametri è ampio, consentendo di esplorare una gamma diversificata di configurazioni in un tempo significativamente ridotto rispetto a una ricerca esaustiva. Analogamente a **GridSearchCV**, anche **RandomizedSearchCV** utilizza la validazione incrociata per valutare le prestazioni del modello su ciascuna combinazione di iperparametri, garantendo una scelta robusta dei parametri ottimali. Tuttavia, grazie alla selezione casuale delle combinazioni, **RandomizedSearchCV** è in grado di trovare una soluzione vicina a quella ottimale in

meno tempo, bilanciando efficienza e accuratezza. GridSearchCV esegue una ricerca esaustiva su tutte le combinazioni possibili di un insieme predefinito di parametri specificati in una griglia. Per ogni combinazione di parametri, il modello viene addestrato e valutato utilizzando la validazione incrociata (cross-validation). Una volta provveduto a definire ogni iper-parametro da ottimizzare con un elenco di valori da esplorare, GridSearchCV esplora sistematicamente ogni possibile combinazione di iper-parametri, scegliendo la combinazione di parametri col punteggio più alto, selezionandola come la migliore. La decisione di utilizzare RandomizedSearchCV è dovuta a diversi vantaggi:

- **Efficienza:** consente di esplorare solo un sottoinsieme casuale limitato di combinazioni di parametri, consentendo di ridurre drasticamente i tempi di calcolo senza compromettere la qualità dei risultati. Mentre GridSearchCV testa tutte le combinazioni possibili degli iperparametri, il che diventa rapidamente proibitivo quando lo spazio dei parametri è ampio o quando il modello è complesso.
- **Flessibilità:** è possibile testare parametri continui e discreti su un ampio range di valori. Alla stessa maniera, invece, GridSearchCV richiede un elenco di valori discreti per ogni parametro, andando a limitare la granularità della ricerca.
- **Generalizzabilità:** entrambe utilizzano la **validazione incrociata (cross-validation)** per garantire che il modello ottimizzato non sia sovra adattato al dataset di training. RandomizedSearchCV, tuttavia, riesce a garantire una **diversificazione dei test**, permettendo di trovare configurazioni migliori.
- **Bilanciamento tra efficienza ed accuratezza:** RandomizedSearchCV, siccome non affronta tutte le combinazioni, riesce pur sempre a trovare una convergenza che più si avvicini a quella ottimale grazie alla casualità e alla diversificazione delle combinazioni testate. E' particolarmente utile quando il risultato perfetto non è strettamente necessario e si può accettare una soluzione "sufficientemente buona" per risparmiare tempo.

Nell'ambito progettuale, l'utilizzo di RandomizedSearchCV si rivela ideale per esplorare in modo efficace uno spazio degli iperparametri ampio, garantendo un

```
#Definizione dei parametri per RandomizedSearchCV
regressor = LogisticRegression(max_iter=1000)
params = {'penalty': ['l1', 'l2'], 'solver': ['saga', 'liblinear'], 'C': list(np.arange(1, 21))}
#Ricerca dei migliori parametri
nreg = RandomizedSearchCV(regressor, param_distributions=params, scoring='accuracy', n_jobs=-1, cv=10, random_state=42)
```

Figura 5.1: Codice per l'utilizzo di RandomizedSearchCV con il modello Logistic Regression.

buon bilanciamento tra efficienza ed accuratezza. Considerando che il dataset è relativamente limitato, un numero ridotto di iterazioni (`n_iter`) risulta sufficiente per individuare configurazioni ottimali senza dover esplorare tutte le possibili combinazioni, come accadrebbe con tecniche esaustive. Questo approccio consente di ottimizzare i modelli in tempi ridotti, mantenendo prestazioni competitive. In seguito viene spiegato l'utilizzo di RandomizedSearchCV all'interno dei modelli scelti.

Logistic Regression:

Nell'immagine 5.1 viene mostrato il codice di RandomizedSearchCV utilizzato nell'algoritmo di Logistic Regression. Nel seguente modello sono presi in input i seguenti **parametri**:

- **penalty:** specifica il tipo di regolarizzazione da utilizzare. 'l1' (Lasso) può portare a modelli più semplici con alcune features a zero, mentre 'l2' (Ridge) penalizza i grandi coefficienti in modo diverso;
- **solver:** ['saga', 'liblinear']: il solver determina l'algoritmo di ottimizzazione usato per la regressione logistica. 'saga' è versatile e può gestire sia L1 che L2, mentre 'liblinear' è ottimizzato per dataset piccoli o medi, è più veloce per problemi binari e supporta solo L1 e L2 per la regressione logistica binaria;
- **C: list(np.arange(1, 21)):** questo parametro controlla la forza della regolarizzazione inversa. Valori più alti di C significano una regolarizzazione più debole (modello meno vincolato), mentre valori più bassi indicano una regolarizzazione più forte. Qui si sta testando un range da 1 a 20.

Vengono inoltre definiti:

```

● ● ●
params = {'criterion': ['gini', 'entropy'], 'min_samples_split': list(np.arange(2, 51)), 'min_samples_leaf': list(np.arange(2, 51))}
# Creo un oggetto di tipo RandomizedSearchCV per cercare i parametri migliori per l'albero
ndt = RandomizedSearchCV(dt, param_distributions=params, cv=10, n_jobs=-1, scoring='accuracy')

```

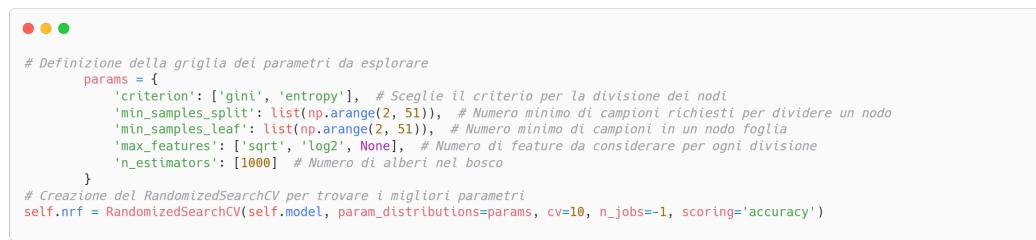
Figura 5.2: Codice per l'utilizzo di RandomizedSearchCV con il modello Decision Tree.

- **regressor:** il modello LogisticRegression da ottimizzare;
- **param_distributions:** i parametri da testare definiti precedentemente;
- **scoring='accuracy':** usa l'accuratezza come metrica per valutare i modelli;
- **n_jobs=-1:** usa tutti i core di CPU disponibili per parallelizzare la ricerca;
- **cv=10:** usa la cross-validation con 10 fold per una stima più robusta della performance;
- **random_state=42:** per garantire la riproducibilità della ricerca casuale.

Decision Tree:

Nell'immagine 5.2 viene mostrato il codice dell'utilizzo di RandomizedSearchCV nell'algoritmo di Decision Tree. Gli input utilizzati sono:

- **dt:** rappresenta l'oggetto DecisionTreeClassifier da ottimizzare;
- **param_distributions=params:** dizionario dei parametri e i rispettivi valori da analizzare;
 - **criterion:** funzione di qualità da usare (Gini Impurity o entropy);
 - **min_samples_split:** numero minimo di campioni richiesti per dividere un nodo (valori da 2 a 50);
 - **min_samples_leaf:** numero minimo di campioni richiesti in una foglia (valori da 2 a 50);
- **cv=10:** numero di fold della cross-validation, utilizzato per validare i risultati;
- **n_jobs=-1:** usa tutti i processori disponibili per parallelizzare il processo;



```

# Definizione della griglia dei parametri da esplorare
params = {
    'criterion': ['gini', 'entropy'], # Sceglie il criterio per la divisione dei nodi
    'min_samples_split': list(np.arange(2, 51)), # Numero minimo di campioni richiesti per dividere un nodo
    'min_samples_leaf': list(np.arange(2, 51)), # Numero minimo di campioni in un nodo foglia
    'max_features': ['sqrt', 'log2', None], # Numero di feature da considerare per ogni divisione
    'n_estimators': [1000] # Numero di alberi nel bosco
}
# Creazione del RandomizedSearchCV per trovare i migliori parametri
self.nrf = RandomizedSearchCV(self.model, param_distributions=params, cv=10, n_jobs=-1, scoring='accuracy')

```

Figura 5.3: Codice per l'utilizzo di RandomizedSearchCV con il modello Random Forest.

- **scoring='accuracy'**: metodologia di scoring utilizzata per valutare le prestazioni del modello.

Random Forest:

Nell'immagine 5.3 viene mostrato il codice di RandomizedSearchCV nell'algoritmo di Random Forest. Gli input sono:

- **self.model**: RandomForestClassifier da ottimizzare;
- **param_distributions=params**: dizionario dei parametri e i rispettivi valori da analizzare:
 - **criterion**: funzione di qualità da usare (Gini o entropy);
 - **min_samples_split**: numero minimo di campioni richiesti per dividere un nodo (valori da 2 a 50);
 - **min_samples_leaf**: numero minimo di campioni richiesti in una foglia (valori da 2 a 50);
 - **max_features**: numero delle features da considerare per ogni divisione;
 - **n_estimators**: numero di alberi nel bosco;
- **cv=10**: numero di fold della cross-validation, utilizzato per validare i risultati;
- **n_jobs=-1**: usa tutti i processori disponibili per parallelizzare il processo;
- **scoring='accuracy'**: metodologia di scoring utilizzata per valutare le prestazioni del modello.

5.2 Logistic Regression

La **Logistic Regression** è uno strumento potente e ampiamente utilizzato in statistica e machine learning per risolvere problemi di classificazione binaria; questo si limita a modellare relazioni continue e la logistic regression è progettata per prevedere probabilità. Questo è particolarmente utile in applicazioni come la diagnosi del diabete, dove il risultato atteso è binario (es. diabete sì/no). Al centro del modello si rintraccia una funzione sigmoide che traduce i valori della combinazione lineare delle variabili indipendenti in un intervallo compreso tra 0 e 1. Questo approccio è stato utilizzato per sviluppare un modello predittivo basato su variabili derivate e trasformate come "Age_Category" (che rappresenta gruppi di età discreti) e "Normalized_Age" (una trasformazione continua e standardizzata dell'età). Tuttavia, la logistic regression può mostrare delle limitazioni quando le relazioni tra le variabili predittive e la classe target non sono lineari o quando sono presenti valori anomali nel dataset. Per affrontare queste sfide sono state utilizzate tecniche di ottimizzazione come RandomizedSearchCV per affinare iper-parametri chiave (ad esempio, la regolarizzazione L1/L2 e il solver). Ciò ha permesso di ottenere un modello robusto e ben adattato ai dati. Questo processo di trasformazione delle probabilità consente alla regressione logistica di modellare scenari in cui l'output non può essere adeguatamente descritto da una relazione puramente lineare. **Motivazioni della scelta di tale modello:**

- **semplicità ed interpretabilità:** la regressione logistica si basa su un modello lineare sottostante facile da comprendere. Si trasforma il risultato lineare con la funzione logistica (o sigmoide) per ottenere una probabilità, ma l'interpretazione dei coefficienti rimane relativamente semplice;
- **efficienza:** richiede relativamente poche risorse computazionali, dove i calcoli necessari per trovare i parametri ottimali sono efficienti, il modello può essere scalato per gestire grandi volumi di dati, mantenendo comunque un'efficienza computazionale;
- **adatto per problemi di classificazione binaria:** è progettato proprio per i problemi di classificazione binaria, dove l'obiettivo è assegnare un'osservazione

```

● ● ●

#Addestramento del modello
nreg.fit(self.X_train, self.y_train)
# Miglior modello
best_model = nreg.best_estimator_
# Calcolo delle metriche
pred_train=nreg.predict(self.X_train)
pred_test = nreg.predict(self.X_test)
# Stampa il miglior parametro
print(nreg.best_params_)
# Stampa l'accuratezza
print("Accuracy:", nreg.best_score_)
# Controllo che il modello migliore non sia None
if best_model is None:
    raise ValueError("Il modello migliore non è stato trovato.")

```

Figura 5.4: Codice per l'utilizzo del modello Logistic Regression.

a una di due categorie

Nella foto 5.4 viene illustrato il codice per utilizzare il modello Logistic Regression, dopo aver preso in input n_reg (calcolato con l'utilizzo di RandomizedSearchCV) viene addestrato il modello con i dati di train e test. Successivamente viene calcolato il miglior modello in maniera da poter poi calcolare le metriche sul modello trovato. Vengono stampati l'accuracy e il miglior parametro del modello ricercato. Nel caso non venisse trovato il miglior modello, viene stampato un messaggio di errore.

5.3 Decision Tree

Il **Decision Tree** è un algoritmo di apprendimento supervisionato utilizzato sia per la classificazione che per la regressione. Tali modelli si basano su una struttura ad albero dove ogni nodo interno rappresenta un'operazione basata su un attributo del dataset e i nodi foglia rappresentano possibili classi o valori di output. Un Decision Tree opera segmentando iterativamente il dataset in sottoinsiemi più piccoli, seguendo un processo di suddivisione che si basa sulla selezione degli attributi più appropriati a ciascun livello. Una delle caratteristiche più importanti di tali modelli è

la loro interpretabilità definita da ramificazioni logiche che consentono di visualizzare i criteri eseguiti dal modello per giungere a una specifica previsione. Un Decision Tree è una struttura ad albero che consiste in:

- **Nodo radice:** punto di partenza dell’albero;
- **Nodi interni:** rappresentano le decisioni o le divisioni sui valori delle caratteristiche di input;
- **Nodi foglia:** rappresentano le classi o i valori previsti.

Componenti fondamentali della costruzione:

1. **Selezione attributo radice:** il primo step per la costruzione si basa sull’identificare l’attributo più rilevante per iniziare la suddivisione dei dati; ciò avviene utilizzando metriche come l’entropia e il guadagno d’informazione, o alternative come il rapporto di guadagno e l’indice di Gini Impurity;
2. **Suddivisione del dataset:** dopo aver scelto l’attributo migliore, i dati vengono divisi in sottogruppi in base ai valori univoci dell’attributo stesso e ciascun sottoinsieme è definito come un ramo dell’albero;
3. **Ricorsione:** per ciascun sottoinsieme si ripete il processo di suddivisione, identificando ulteriori attributi chiave, fino a raggiungere nodi foglia o soddisfare criteri di stop predefiniti, come la profondità massima o una dimensione minima del sottoinsieme.

Motivazione:

- **facilità di interpretazione e spiegazione:** gli alberi decisionali sono semplici da capire e visualizzare. Ogni decisione o previsione può essere facilmente tracciata attraverso il percorso dell’albero, rendendoli ideali quando è importante spiegare le decisioni ai non esperti;
- **trattamento di dati categorici e continui:** sono flessibili nel trattare sia dati categorici che continui senza la necessità di particolari trasformazioni;

```

● ● ●

# Creo l'oggetto di tipo DecisionTreeClassifier bilanciato sui pesi
clf = DecisionTreeClassifier(class_weight='balanced')
# Addestra il Decision Tree con i dati di x_t e y_t
clf = clf.fit(self.x_t, self.y_t)
# Il modello predice le etichette per i dati di test
y_pred = clf.predict(self.x_te)
# Calcolo dell'accuratezza
print("Accuracy:", metrics.accuracy_score(self.y_te, y_pred))

```

Figura 5.5: Codice per l'utilizzo del modello Decision Tree senza l'utilizzo di RandomizedSearchCV.

- **richiedono meno preparazione dei dati:** a differenza di altri modelli come le reti neurali, i Decision Tree richiedono meno preprocessing dei dati (ad esempio, non richiedono la normalizzazione degli input);
- **velocità ed efficienza:** possono essere addestrati rapidamente e sono efficienti nei calcoli rispetto a modelli più complessi come Support Vector Machines o reti neurali profonde;
- **overfitting:** gli alberi possono adattarsi troppo ai dati di training, perdendo generalizzazione. Tecniche di potatura (pruning) o approcci come gli ensemble methods (ad esempio, Random Forest) sono spesso necessari per mitigare questo problema.

Lo spezzone di codice figura 5.5 mostra la creazione dell'oggetto DecisionTreeClassifier, l'addestramento del modello sui dati di test e train, successivamente predice le etichette per i dati ed, infine, stampa l'accuracy del modello. Il seguente caso è stato svolto per mostrare la differenza del Decision Tree con e senza l'utilizzo di RandomizedSearchCV. In questo caso viene scelto il primo modello e viene creato l'albero. Essendo che viene selezionato il primo modello, l'albero avrà una dimensione elevata. L'albero creato con questo metodo viene mostrato nella figura 5.7.

```

● ● ●
# Addestra il modello sui dati forniti
ndt.fit(self.x_t, self.y_t)
# Stampa il miglior parametro
print(ndt.best_params_)
# Stampa l'accuracy
print("Accuracy:", ndt.best_score_)

```

Figura 5.6: Codice per l'utilizzo del modello Decision Tree con l'utilizzo di RandomizedSearchCV.

Nel codice della figura 5.6, dopo aver preso in input il modello calcolato con l'utilizzo di RandomizedSearchCV, addestra il modello con i dati di test e train, stampa il miglior parametro e l'accuracy del modello. Questa volta l'albero che viene creato sarà di dimensioni più piccole rispetto a quello precedente, in quanto con RadomizedSearchCV viene scelto il modello migliore. L'albero creato con questo metodo viene mostrato nella figura 5.8.

5.4 Random Forest

Il **Random Forest** rappresenta un avanzato algoritmo di apprendimento automatico che si basa sulla costruzione e sull'aggregazione di una collezione di alberi decisionali (decision trees). Attraverso il principio dell'ensemble learning, questo modello combina le predizioni di numerosi alberi decisionali individuali per ottenere una stima finale più accurata e robusta, mitigando significativamente il rischio di overfitting. Il Random Forest integra due tecniche fondamentali: il **bagging** (bootstrap aggregating), che crea sottoinsiemi casuali dei dati di addestramento per ciascun albero, e la **selezione casuale** delle caratteristiche, che per ogni nodo di divisione seleziona un sottoinsieme casuale di variabili da considerare, promuovendo così la diversità tra gli alberi e riducendo la varianza complessiva del modello.

Il Random Forest è molto rilevante per i seguenti aspetti:

- **Analisi della feature importance:** la capacità del Random Forest di determinare l'importanza delle variabili, agevolando l'identificazione di quei fattori che influenzano maggiormente i risultati predetti. Questo non solo rende il modello più interpretabile, ma offre anche indicazioni utili per orientare studi futuri,

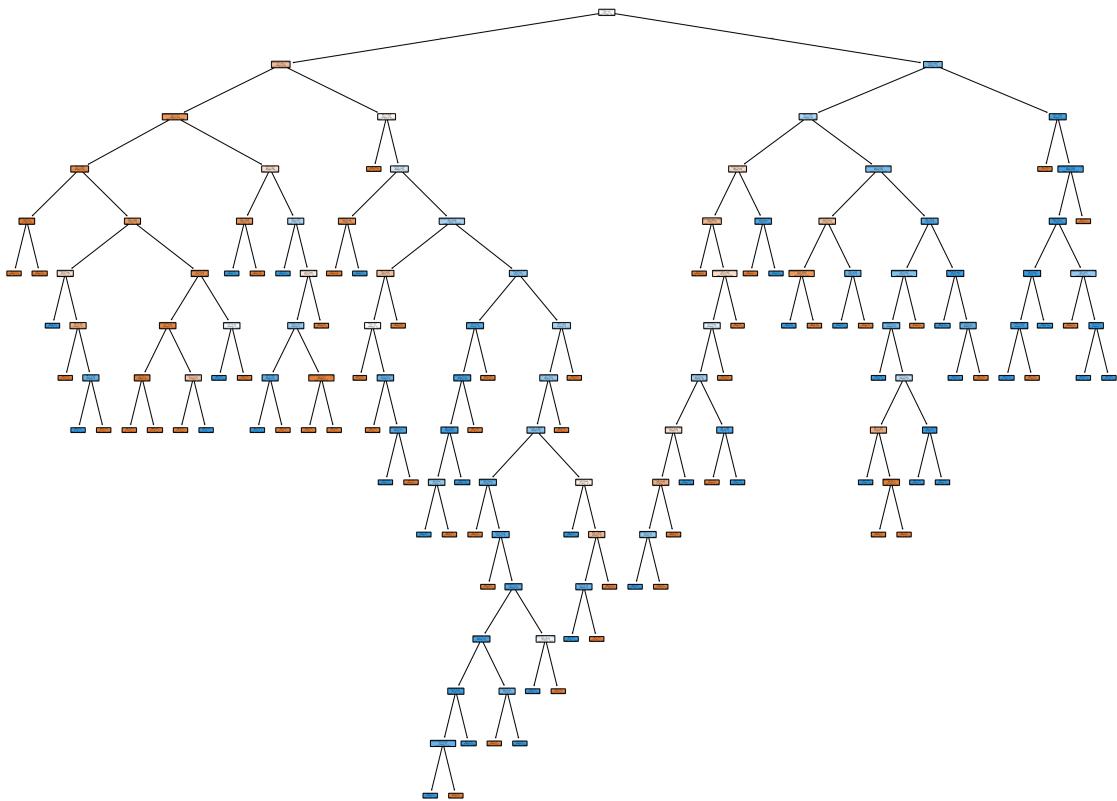


Figura 5.7: Decision Tree senza l'utilizzo di RandomizedSearchCV.

focalizzandosi sulle variabili essenziali per una migliore comprensione del fenomeno analizzato;

- **Ottimizzazione della generalizzazione:** la strategia di ensemble adottata dal Random Forest contribuisce a migliorare l'abilità del modello nel generalizzare le predizioni su dati non inclusi nell'addestramento. Grazie al fatto che ogni albero decisionale viene costruito su un campione diverso e utilizza un sottoinsieme specifico delle variabili, si riduce la correlazione tra gli alberi e, di conseguenza, la varianza complessiva del modello;
- **Robustezza agli outlier:** nei dati clinici, è comune riscontrare outlier, come valori estremi di biomarcatori quali glucosio o insulina. La struttura di ensemble del Random Forest offre una notevole robustezza a tali anomalie. Poiché ogni albero considera solo una parte dei dati complessivi, il peso dei valori atipici viene attenuato, rendendo le previsioni più affidabili e meno suscettibili alle distorsioni causate dagli outlier;

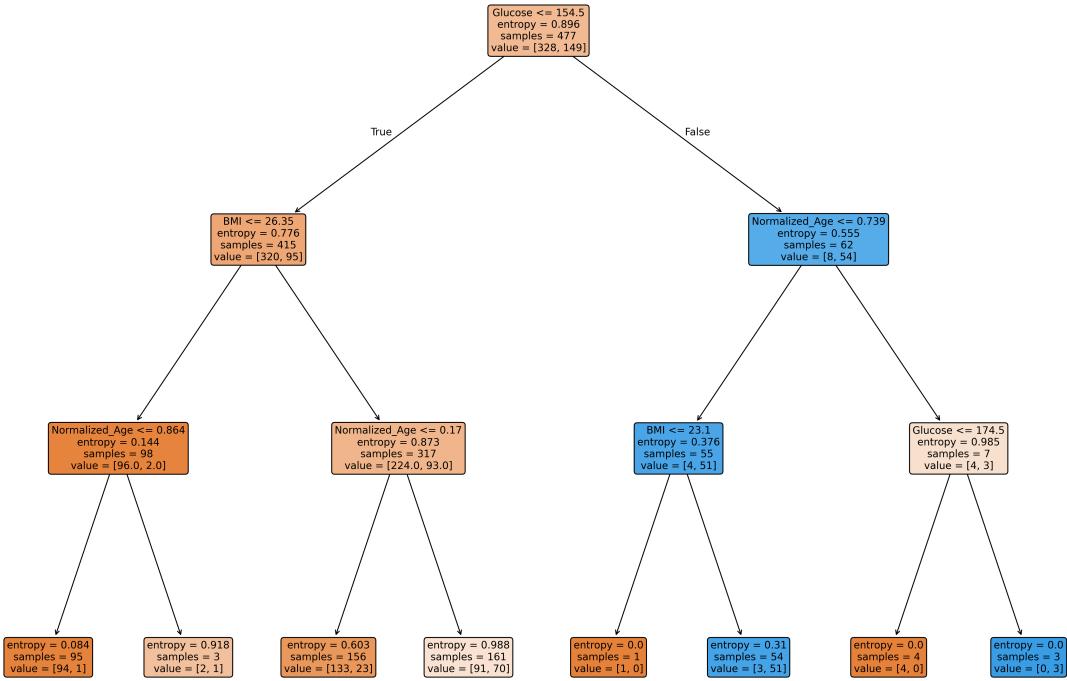


Figura 5.8: Decision Tree con l'utilizzo di RandomizedSearchCV.

- **Gestione dei dati incompleti:** è la capacità intrinseca di gestire i dati mancanti senza dover necessariamente ricorrere a tecniche esterne di imputazione. L'abilità di lavorare con insiemi parzialmente completi consente analisi più robuste e meno influenzate da eventuali bias derivanti dalla gestione dei valori mancanti.

Il codice di figura 5.9 dopo aver calcolato il miglior modello con l'utilizzo di RandomizedSearchCV viene addestrato il modello sui dati di test e train, vengono poi stampati il miglior parametro e l'accuracy del modello. Così come per il Decision Tree anche con il Random Forest viene creato un albero di decisione in quanto Random Forest utilizza l'ensemble learning, che combina le previsioni di più modelli (alberi decisionali) per produrre una previsione finale più robusta e accurata. L'albero creato con il modello Random Forest è illustrato nella figura 5.10

```

● ● ●

# Allena il modello con i dati di addestramento
self.nrf.fit(self.x_train, self.y_train)
# Stampa i migliori parametri e il miglior punteggio
print("Best Parameters:", self.nrf.best_params_)
print("Best Score:", self.nrf.best_score_)
    
```

Figura 5.9: Codice per l'utilizzo del modello Random Forest.

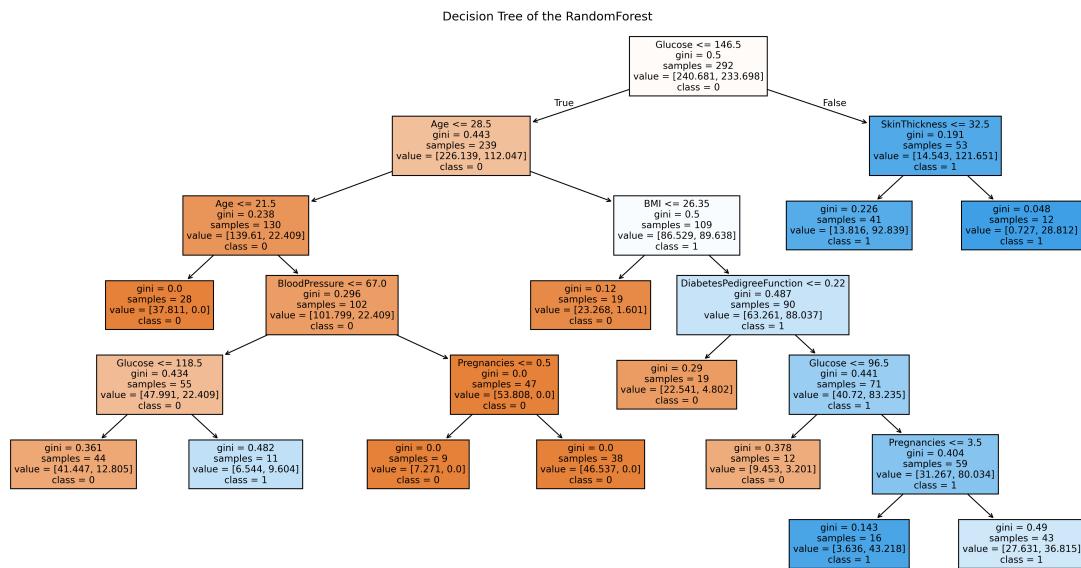


Figura 5.10: Modello Random Forest.

CAPITOLO 6

Valutazione dei risultati

I risultati dei diversi modelli di Machine Learning sono presentati di seguito, insieme a un’analisi delle loro prestazioni sia nella fase di testing che sul set di test. Viene fornita una panoramica delle matrici di confusione per ciascun modello, consentendo un’analisi dettagliata non solo dell’accuratezza complessiva, ma anche della capacità di ogni modello di distinguere correttamente tra campioni diabetici e non, con particolare enfasi sulla minimizzazione dei falsi negativi.

6.1 Confusion Matrix

Una **matrice di confusione** riassume le istanze di test in base ai valori predetti e ai valori veri e sono presentati come tabella di contingenza. La confusion matrix è organizzata nella seguente maniera:

- **True Positive (TP)**: casi di previsione corretta di una **classe positiva**;
- **True Negative (TN)**: casi in cui il modello ha correttamente predetto una **classe negativa**;
- **False Positive (FP)**: casi in cui il modello ha **erroneamente** predetto una classe positiva, quando in realtà era negativa;

- **False Negative (FN):** casi in cui il modello ha erroneamente predetto una classe negativa quando in realtà era positiva. Si tratta quindi di una **mancata identificazione**.

6.2 Metriche di valutazione

Le metriche delle performance del machine learning vengono utilizzate per valutare e misurare le prestazioni di un modello predittivo. Gli indicatori di performance sono fondamentali, essendo che possano consentire di valutare l'efficacia del modello predittivo nella risoluzione di una determinata problematica e fornire una base oggettiva per il confronto di approcci diversificati. Esistono diversi e vari tipi di metriche delle performance di ML, ciascuna delle quali fornisce un'importante prospettiva sulle performance di un modello di machine learning.

- **Precision:** è la metrica di valutazione più semplice. Indica la percentuale di predizioni corrette di classe positiva rispetto al totale delle predizioni di classe positiva;

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall:** indica la percentuale di predizioni corrette di classe positiva rispetto al totale dei casi di classe positiva. È anche chiamata **True Positive Rate**;

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **Accuracy:** è una misura più generale sulle performance del modello. Essa indica il numero di volte che il modello ha classificato correttamente rispetto al numero totale di occorrenze da classificare;

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **F1_Score:** considera sia la Precision che la Recall ed equivale alla loro media armonica.

$$\text{F1_Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Modello	Accuracy	Precision	Recall	F1_Score
Logistic Regression	0.8050314465408805	0.7592592592592593	0.5503355704697986	0.6381322957198443

Figura 6.1: Metriche relative ai dati di training sul modello Logistic Regression.

6.3 Curva AUC ROC

L’**Area sotto la curva (AUC)** della **receiver operating characteristic (ROC)** rappresenta una misura consolidata e ampiamente utilizzata nella classificazione binaria. La curva ROC è un grafico che mette in relazione il tasso di veri positivi con il tasso di falsi positivi, calcolati a diverse soglie di probabilità che variano da 0 a 1. In dettaglio:

- **il True Positive Rate (TPR):** anche detto recall, misura la proporzione di positivi correttamente identificati del modello rispetto al totale dei positivi reali;
- **il False Positive Rate (FPR):** rappresenta la proporzione di negativi erroneamente classificati come positivi rispetto al totale dei negativi reali.

6.4 Logistic Regression

Nella figura 6.1 sono riportate le metriche per il modello predittivo Logistic Regression, i dati di training del miglior modello, calcolato grazie all’uso di RandomizedSearchCV ha riportato **un’accuratezza di 0.805 con un F1_Score di 0.638**. L’accuratezza riportata da questo modello risulta essere la più bassa rispetto agli altri modelli, anche se la differenza risulta minima.

La matrice di confusione, sui dati di training, riportata in figura 6.2 rivela che il modello ha correttamente classificato la classe **negativa con 302 campioni** rispetto agli **82 campioni della classe positiva**, inoltre sono presenti **26 falsi positivi e 67 falsi negativi**. Con il modello di Logistic Regression sono state ottenute buone performance anche se la presenza dei falsi negativi indica che ci sono stati casi in cui chi aveva il diabete risultasse negativo. E’ importante considerare questo aspetto, in quanto per il diabete è previsto un rigido sistema di alimentazione e quindi persone che sono falsi negativi potrebbero avere dei seri problemi.

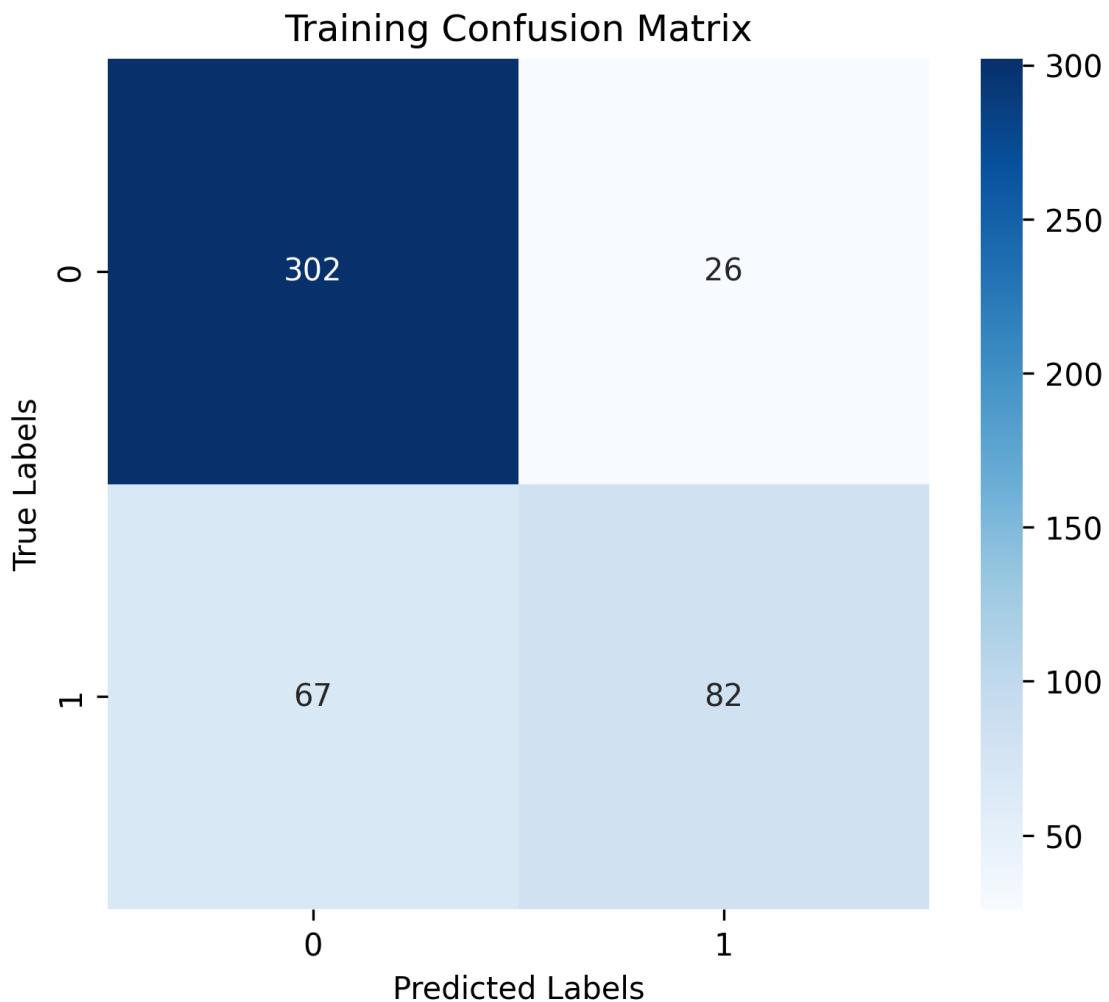


Figura 6.2: Confusion Matrix del modello Logistic Regression sui dati di training.

Le metriche di figura 6.3, per il modello predittivo Logistic Regression sui dati di test del miglior modello, calcolato grazie all'uso di RandomizedSearchCV ha riportato **un'accuratezza di 0.748 con un F1_Score di 0.556**. L'accuratezza riportata da questo modello risulta essere quella intermedia.

La matrice di confusione figura 6.4, sui dati di test, rivela che il modello ha correttamente classificato la classe **negativa con 94 campioni** rispetto ai **25 campioni della classe positiva**, inoltre sono presenti **17 falsi positivi e 23 falsi negativi**.

Modello	Accuracy	Precision	Recall	F1_Score
Logistic Regression	0.7484276729559748	0.5952380952380952	0.5208333333333334	0.5555555555555556

Figura 6.3: Metriche relative ai dati di test sul modello Logistic Regression.

6.5 Decision Tree

I dati riportati nella figura 6.6 si riferiscono al miglior modello del Decision Tree, che sui dati di training, ha rilevato un **accuracy di 0.830 ed un F1_Score di 0.682**. L'accuracy calcolata risulta essere la migliore tra i modelli utilizzati. Questo vuol dire che il Decision Tree è riuscito a classificare al meglio i dati di training.

La matrice di confusione di figura 6.7, sui dati di training, rivela che il modello ha correttamente classificato la classe **negativa con 309 campioni** rispetto agli **87 campioni della classe positiva**, inoltre sono presenti **19 falsi positivi e 62 falsi negativi**. Con il modello Decision Tree sono state ottenute buone performance. A differenza della matrice di confusione della Logistic Regression sui dati di training, possiamo notare che in questo caso i veri negativi sono maggiori, quindi questo modello è riuscito a classificare al meglio la classe negativa. Inoltre il numero di falsi negativi risulta essere minore per il Decision Tree, quindi se lo scopo è diminuire i falsi negativi questo modello risulta essere migliore.

I dati di figura 6.8 si riferiscono al miglior modello del Decision Tree, che sui dati di test, ha rilevato un **accuracy di 0.767 ed un F1_Score di 0.519**. L'accuracy calcolata anche in questo caso risulta essere la migliore tra i modelli utilizzati. Questo vuol dire che il Decision Tree sui dati di training e di test è riuscito a classificare al meglio i dati.

La matrice di confusione di figura 6.9 sui dati di test mostra che il modello ha classificato correttamente **102 campioni nella classe negativa** rispetto ai **20 campioni della classe positiva**. Tuttavia, ci sono **9 falsi positivi e 28 falsi negativi**. Confrontando questi risultati con la matrice di confusione della Logistic Regression sui dati di test, notiamo che questo modello ha un numero maggiore di veri negativi, dimostrando una migliore capacità di classificazione della classe negativa. Tuttavia, il Decision Tree presenta un numero maggiore di falsi negativi. Pertanto, se l'obiettivo è ridurre

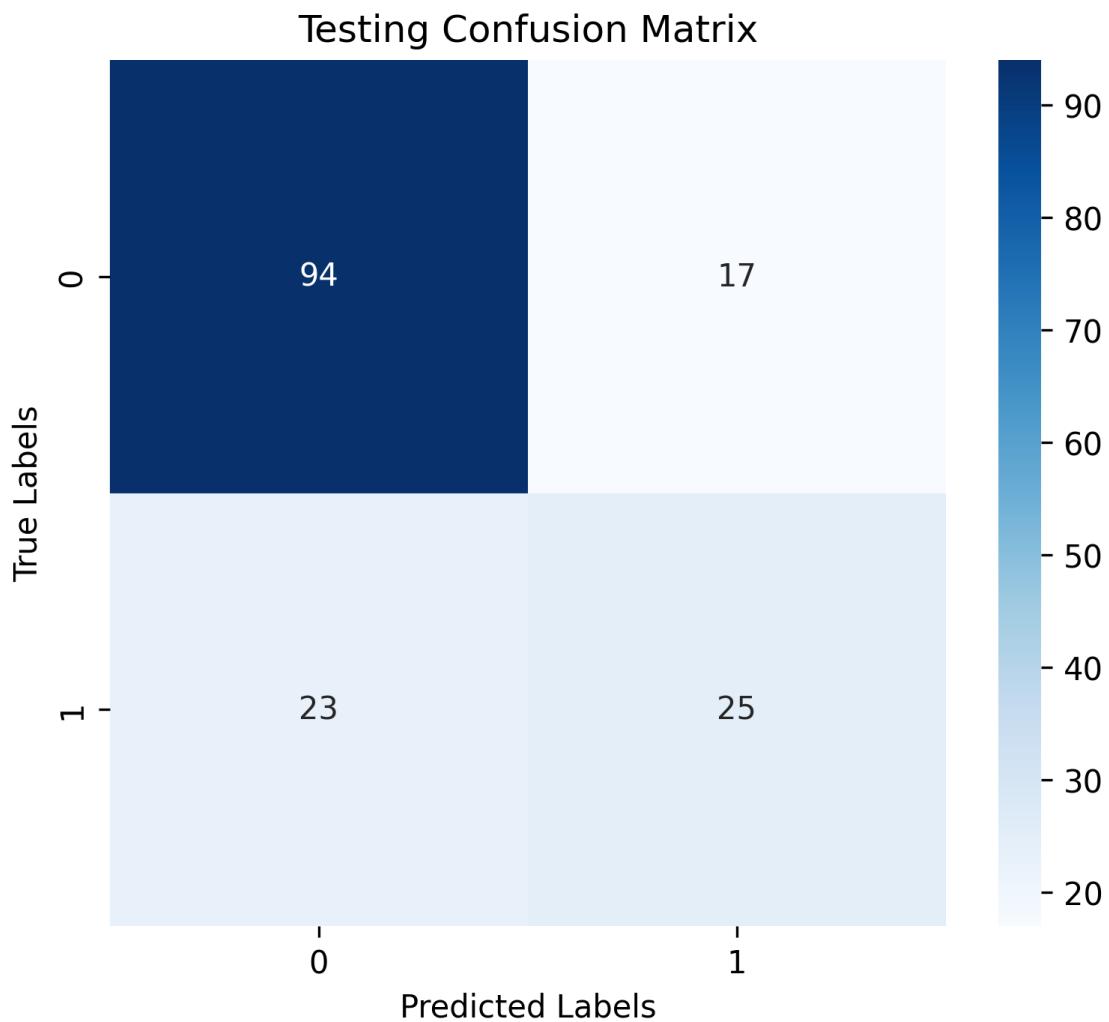


Figura 6.4: Confusion Matrix del modello Logistic Regression sui dati di test.

i falsi negativi, la Logistic Regression risulta essere una scelta migliore.

6.6 Random Forest

Le metriche di figura 6.11 sul miglior modello del Random Forest, sui dati di training, ha rilevato un **accuracy di 0.825 ed un F1_Score di 0.755**. L'accuracy ottenuta per il Random Forest varia di poco rispetto ai precedenti modelli di LogisticRegression e DecisionTree.

La matrice di confusione di figura 6.12, calcolata sui dati di training, mostra che il modello ha classificato correttamente **268 campioni della classe negativa e 126 della classe positiva**. Tuttavia, sono presenti **60 falsi positivi**, che indicano

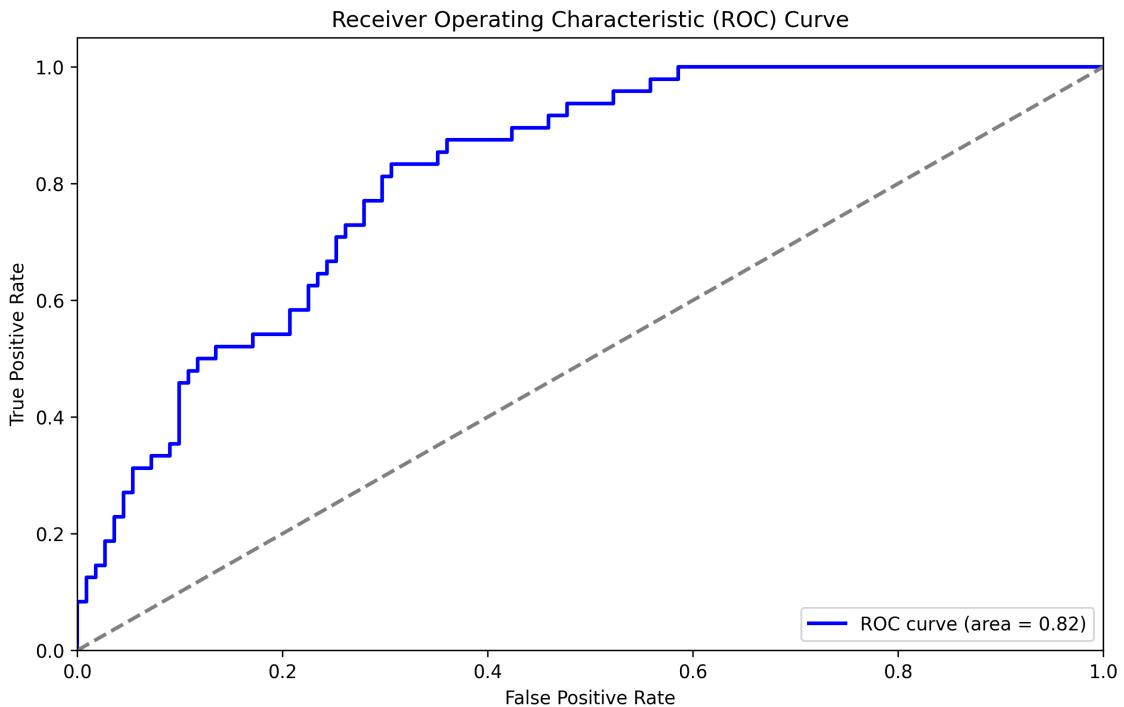


Figura 6.5: ROC Curve del modello Logistic Regression.

Modello	Accuracy	Precision	Recall	F1_Score
Decision Tree	0.8301886792452831	0.8207547169811321	0.5838926174496645	0.6823529411764706

Figura 6.6: Metriche relative ai dati di training sul modello Decision Tree.

un numero maggiore di campioni negativi erroneamente classificati come positivi rispetto alle matrici di confusione della Logistic Regression e del Random Forest. Allo stesso tempo, il numero di **falsi negativi 23** è il più basso tra i modelli analizzati, suggerendo una maggiore efficacia nel rilevare correttamente i campioni positivi. In sintesi, se l’obiettivo primario è ridurre i falsi negativi, questo modello offre una prestazione migliore rispetto agli altri. Tuttavia, il costo di questa ottimizzazione è un aumento dei falsi positivi, che potrebbe non essere ideale in contesti dove la classificazione accurata della classe negativa è fondamentale.

La figura 6.13 riporta le metriche del miglior modello del Random Forest, sui dati di test, ha rilevato un **accuracy di 0.735 ed un F1_Score di 0.625**. L’accuracy calcolata risulta essere la peggiore quindi per i dati di test, il Random Forest non riesce a classificare bene i dati. Tuttavia la F1_Score risulta essere la migliore, quindi

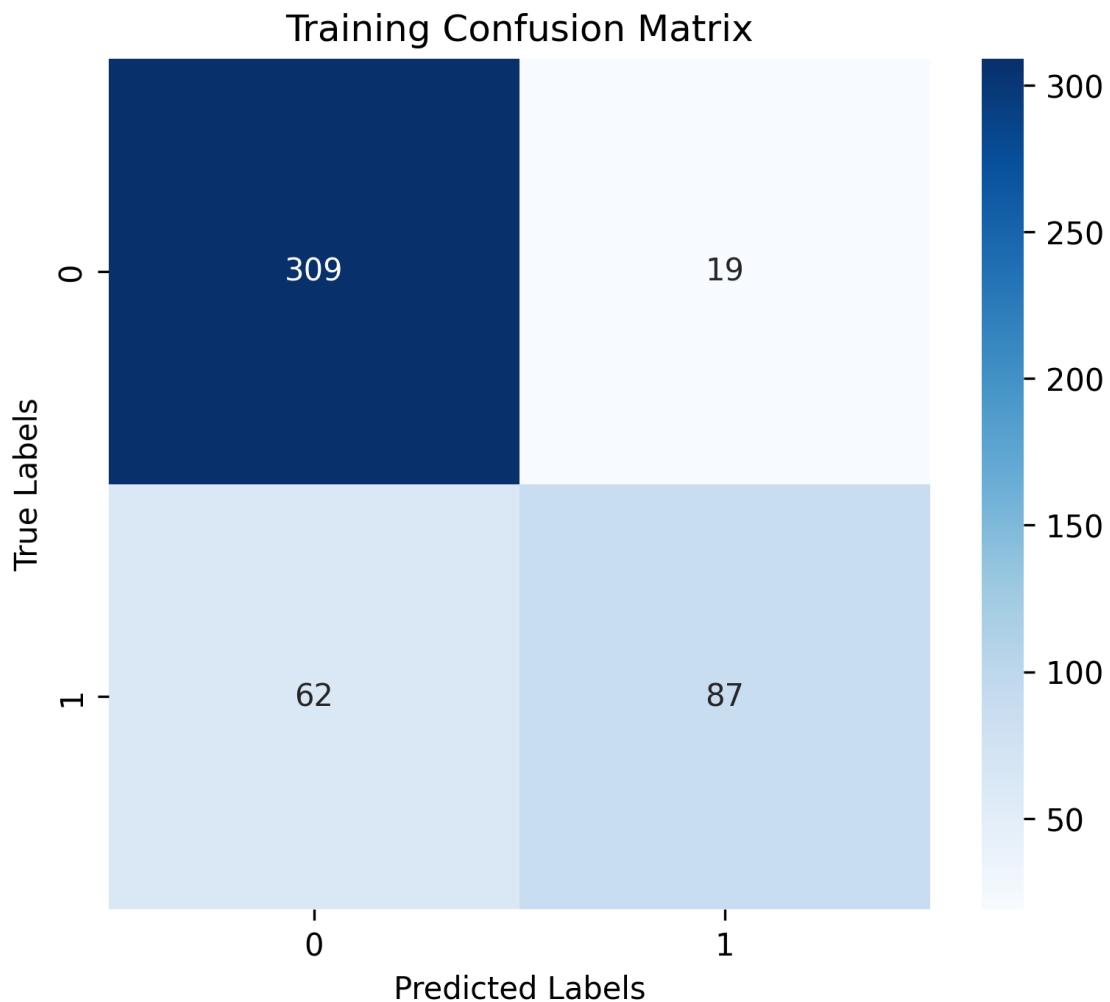


Figura 6.7: Confusion Matrix del modello Decision Tree sui dati di training.

in questo caso il modello bilancia al meglio la precision e la recall.

La matrice di confusione sui dati di test di figura 6.14 mostra che il modello Random Forest ha classificato correttamente **82 campioni nella classe negativa** e **35 campioni nella classe positiva**. Tuttavia, ci sono **29 falsi positivi** e **13 falsi negativi**. Confrontando questi risultati con la matrice di confusione della Logistic Regression e del Decision Tree sui dati di test, notiamo che il Random Forest presenta il **numero più basso di falsi negativi**. Pertanto, rispetto agli altri modelli, il Random Forest è più efficace nel ridurre i falsi negativi.

Modello	Accuracy	Precision	Recall	F1_Score
Decision Tree	0.7672955974842768	0.6896551724137931	0.4166666666666667	0.5194805194805194

Figura 6.8: Metriche relative ai dati di test sul modello Decision Tree.

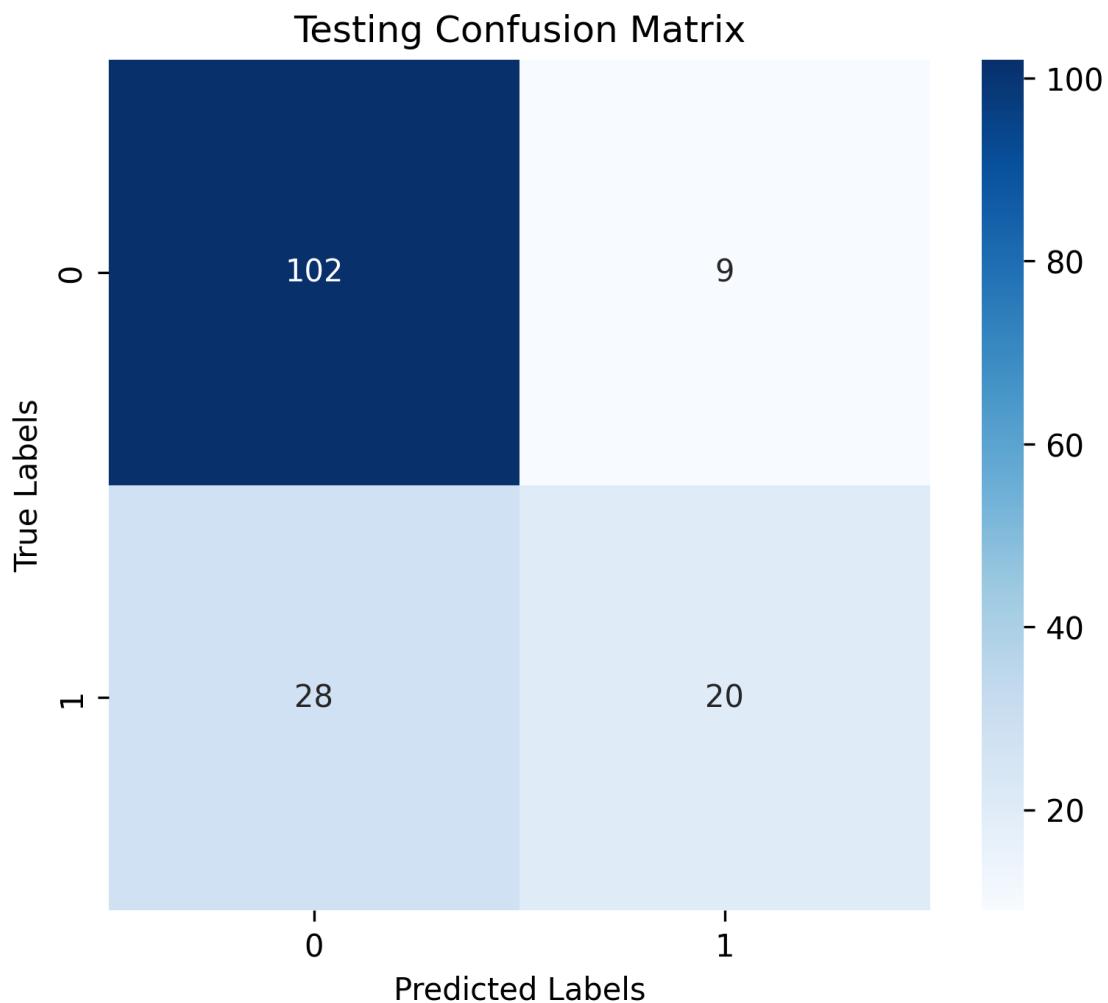


Figura 6.9: Confusion Matrix del modello Decision Tree sui dati di test.

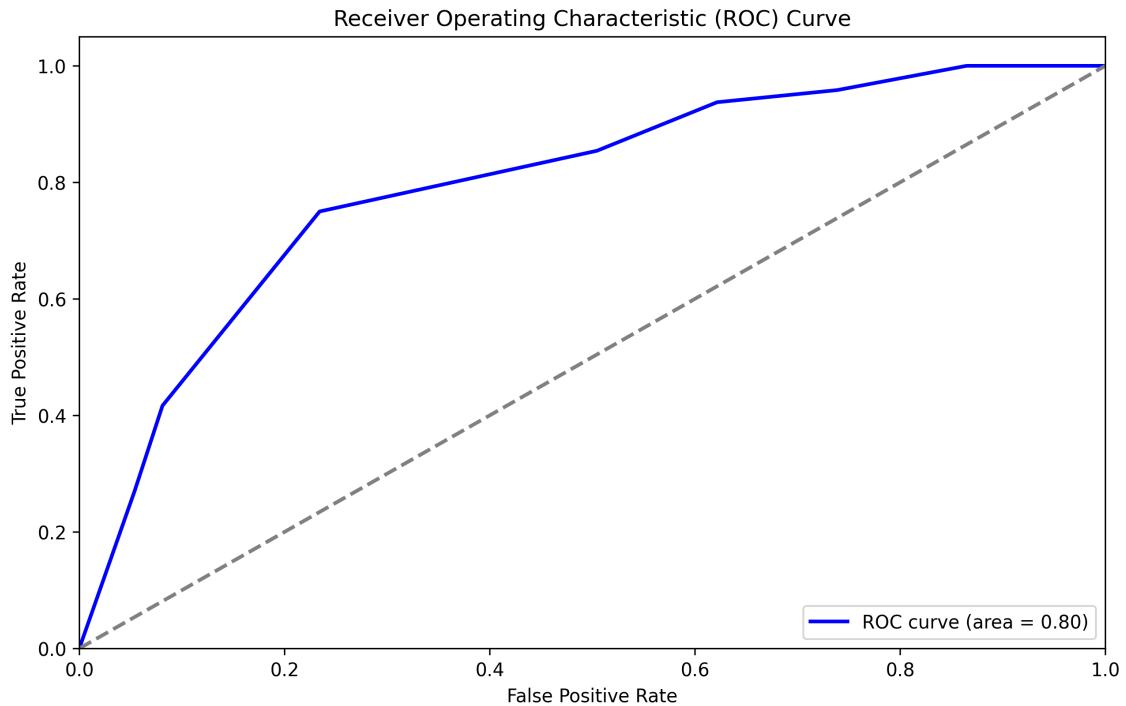


Figura 6.10: ROC Curve del modello Decision Tree.

Modello	Accuracy	Precision	Recall	F1_Score
Random Forest	0.8259958071278826	0.6774193548387096	0.8456375838926175	0.7522388059701492

Figura 6.11: Metriche relative ai dati di training sul modello Random Forest.

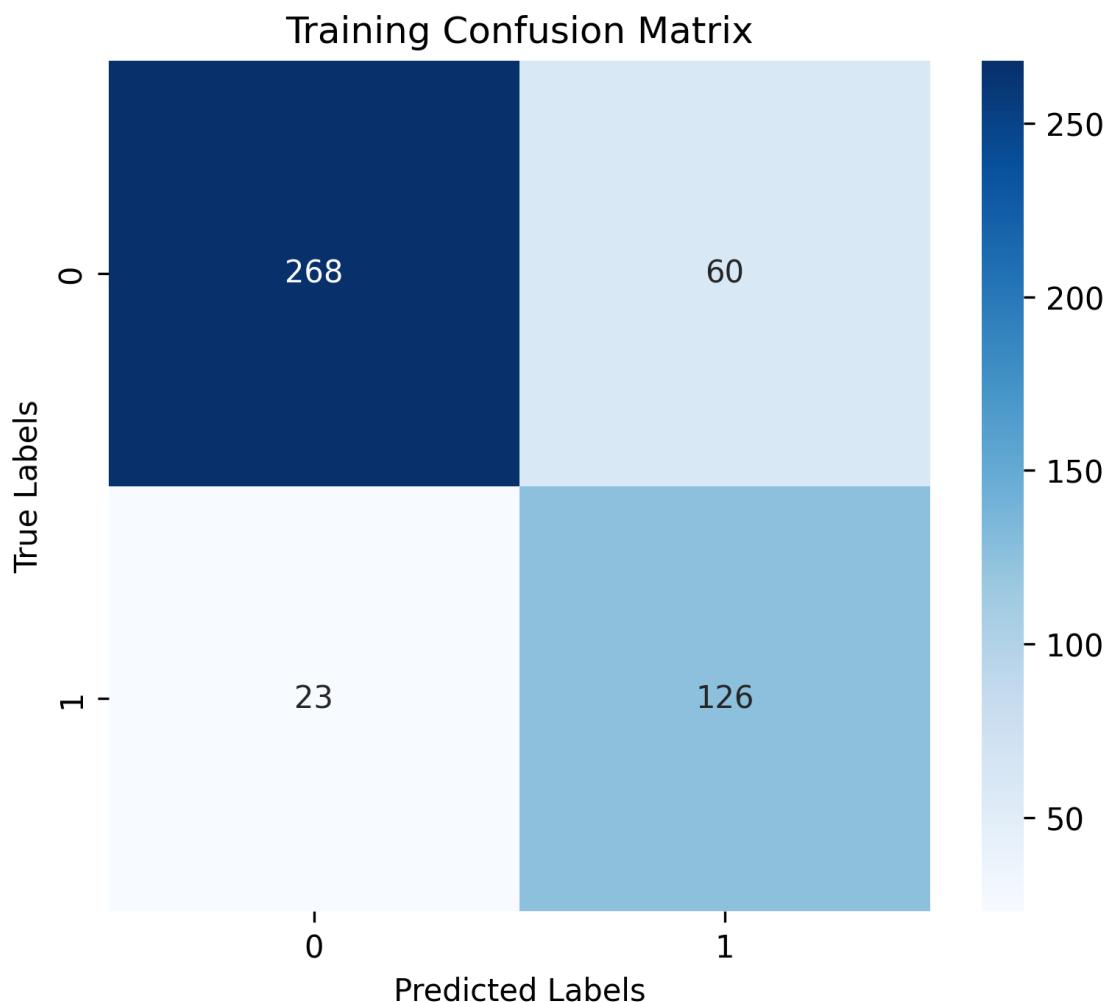


Figura 6.12: Confusion Matrix del modello Random Forest sui dati di training.

Modello	Accuracy	Precision	Recall	F1_Score
Random Forest	0.7358490566037735	0.546875	0.7291666666666666	0.625

Figura 6.13: Metriche relative ai dati di test sul modello Random Forest.

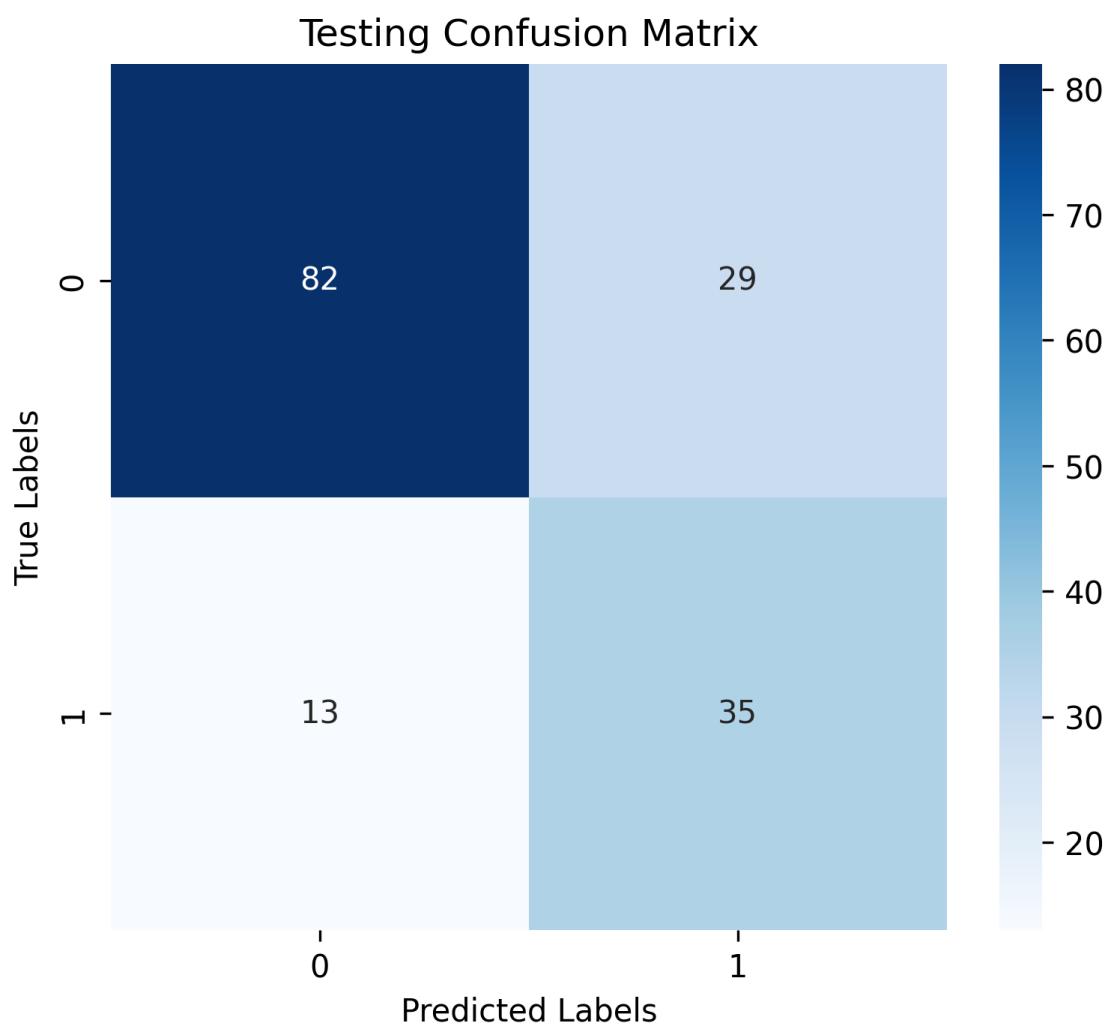


Figura 6.14: Confusion Matrix del modello Random Forest sui dati di test.

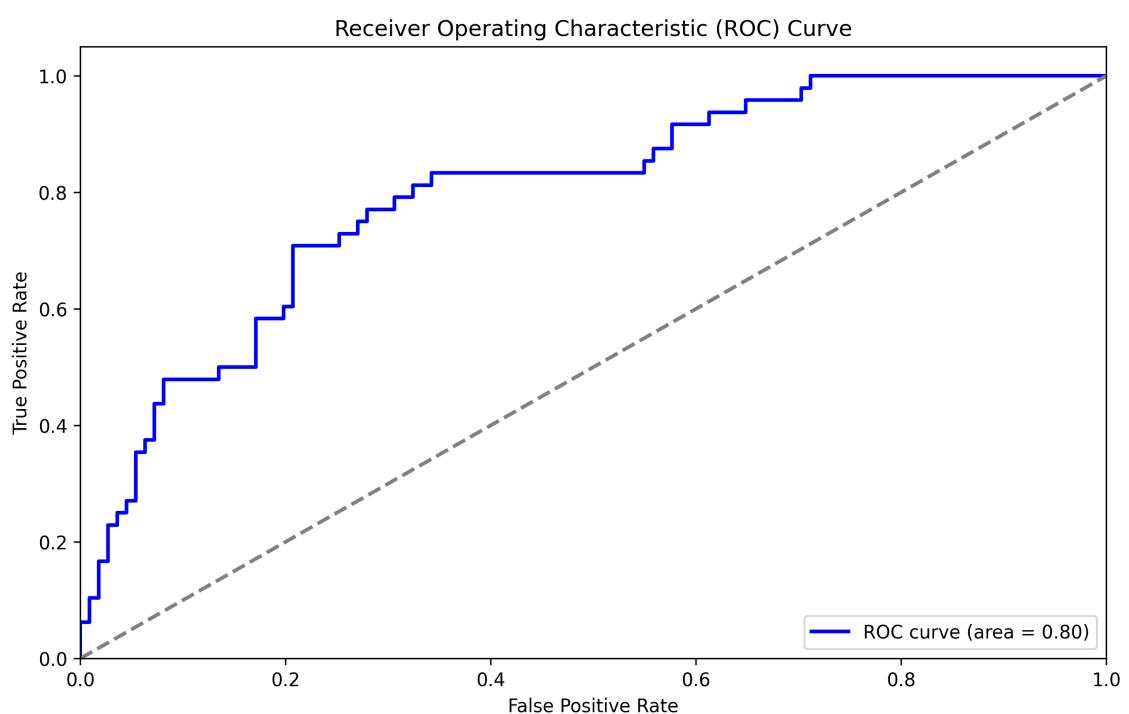


Figura 6.15: ROC Curve del modello Random Forest.

CAPITOLO 7

Conclusioni

Basandosi sulle metriche calcolate sui dati di training e sui dati di test possiamo fornire due conclusioni:

1. Nel primo caso dove è fondamentale ridurre al minimo i **falsi negativi** per diagnosticare i pazienti diabetici, il **Random Forest** risulta essere la scelta migliore sia per i dati di training che i dati di test. Grazie al suo Recall più alto identifica con maggiore sicurezza i pazienti a rischio offrendo un bilanciamento accettabile tra Precision e Recall, rappresentato dal miglior **F1_Score**.
2. Nel secondo caso se si desidera minimizzare i **falsi positivi**, il **Decision Tree** è il miglior modello grazie alla sua Precision più alta, avendo inoltre un'accuratezza elevata che bilancia meglio l'identificazione corretta dei pazienti.

Basandosi esclusivamente sull'AUC delle figure 6.5, 6.10 e 6.15, la **Logistic Regression** è il modello migliore poichè ha il valore più alto (**0.82**), quindi ha una capacità leggermente superiore a distinguere correttamente tra pazienti diabetici e non diabetici.

Bibliografia

- [1] I. S. di Sanità. (2023) Diabete. [Online]. Available: <https://www.epicentro.iss.it/diabete/> (Citato alle pagine 1 e 2)
- [2] E. G. Puffenberger, M. J. Morton, K. H. Strauss, T. L. Hendrickson, K. J. Robinson, R. D. Tierney, J. J. Sunyaev, D. A. Shinde, J. M. Assir, J. Nasir, T. Husnain, M. A. Khalid, M. W. Awan, L. A. Leal, M. F. DiMaria, D. T. Wood, H. H. Kazazian, D. N. Finegold, D. T. Roche, B. H. Lee, S. F. A. Grant, C. W. Tollinson, D. A. Piccione, and H. Hakonarson, "Identification of novel ethnic-specific candidate genes associated with type 2 diabetes," *Frontiers in Genetics*, vol. 9, 2018. [Online]. Available: <https://www.frontiersin.org/articles/10.3389/fgene.2018.00515/full> (Citato a pagina 4)
- [3] F. Nassiwa and J. Zeng, "Evaluating traditional machine learning models for predicting diabetes onset using the pima indians dataset," Available at SSRN 4878052, 2022. (Citato a pagina 7)
- [4] P. S. Moon, P. A. Bainalwar, S. M. Borkar, and S. S. Shambharkar. (2024) Machine learning approach for diabetes prediction using pima dataset. New York, NY, USA. [Online]. Available: <https://doi.org/10.1145/3647444.3652479> (Citato a pagina 8)

- [5] K. Zhao and Z. Wang, "Research on diabetes prediction based on machine learning," in *Proceedings of the 6th International Conference on Machine Learning and Machine Intelligence*, ser. MLMI '23. New York, NY, USA: Association for Computing Machinery, 2024, p. 29–33. [Online]. Available: <https://doi.org/10.1145/3635638.3635643> (Citato a pagina 8)