UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA

Corso di Laurea in Informatica

DIABETES PREDICTOR

Progetto di Fondamenti di Intelligenza Artificiale

Relatore:
Prof.
Fabio Palomba

Candidati:
Luca Pastore
Mat. 0512113201

Giovanni Borrelli
Mat. 0512110177

Agostino Andrea Mangia
Mat. 0512112157

ANNO ACCADEMICO 2022/2023

SOMMARIO

CHAPTER 1

INTRODUZIONE

1.1 Definizione del problema

Il diabete rappresenta una sfida significativa per la salute pubblica a livello globale. La diagnosi tempestiva e accurata del diabete è essenziale per prevenire complicanze gravi e migliorare la qualità della vita dei pazienti. Tuttavia, attualmente la diagnosi del diabete si basa principalmente su test di laboratorio che richiedono tempo e risorse significative.
In questo contesto, l'applicazione di algoritmi di machine learning per la predizione del diabete può fornire un approccio promettente per identificare in modo precoce i pazienti a rischio di sviluppare la malattia. L'obiettivo principale di questo progetto è quello di sviluppare e valutare modelli di machine learning in grado di predire con precisione la presenza o l'insorgenza del diabete.
Il problema principale affrontato in questa tesi riguarda la complessità e l'incertezza legate alla predizione del diabete. La varietà di fattori di rischio e l'interazione complessa tra di essi rendono la predizione del diabete un compito impegnativo. Inoltre, l'utilizzo di dati clinici e diagnostici incompleti o inaccurati può influire sulla precisione e l'affidabilità dei modelli di

predizione.
Sarà necessario esaminare e confrontare diversi algoritmi di machine learning, per determinare quale approccio offre le migliori performance nella predizione del diabete.
Attraverso questa ricerca, si mira a fornire una base scientifica solida per l'implementazione di sistemi di predizione del diabete basati su machine learning, che possano essere utilizzati come strumenti di supporto decisionale per i professionisti sanitari. Inoltre, si prevede che i risultati di questo progetto possano contribuire allo sviluppo di interventi preventivi e personalizzati per individui a rischio di sviluppare il diabete, migliorando così la gestione e il controllo della malattia.

1.2 Motivazioni e Obiettivi

Questo progetto si propone di esplorare l'applicazione dell'intelligenza artificiale nella diagnosi del diabete, sviluppando un sistema diagnostico basato su algoritmi di machine learning. L'obiettivo principale è sviluppare un modello predittivo in grado di identificare il diabete sulla base di dati clinici e di laboratorio, migliorando la precisione diagnostica e riducendo il rischio di diagnosi errate o ritardate.
Il sistema proposto utilizzerà una vasta gamma di variabili cliniche e di laboratorio, come livelli di glucosio nel sangue, età, BMI e altri fattori di rischio noti.
Un aspetto cruciale del lavoro di ricerca sarà la validazione del modello predittivo, valutandone l'accuratezza e l'affidabilità utilizzando dati indipendenti e confrontandolo con metodi diagnostici tradizionali. Saranno effettuati confronti e analisi comparative per valutare le prestazioni del modello e la sua efficacia nel contesto clinico.
L'implementazione di un sistema diagnostico basato sull'intelligenza artificiale per la diagnosi del diabete potrebbe portare a numerosi vantaggi, tra cui una diagnosi più rapida ed efficiente, la riduzione dei costi e una

migliore gestione delle risorse sanitarie.

1.3 CRISP-DM

Il CRISP-DM (Cross-Industry Standard Process for Data Mining) è il modello di ciclo di vita del software standard per progetti di Machine Learning. Si compone di fasi che possono essere eseguite un numero illimitato di volte, il che lo rende un modello non sequenziale.

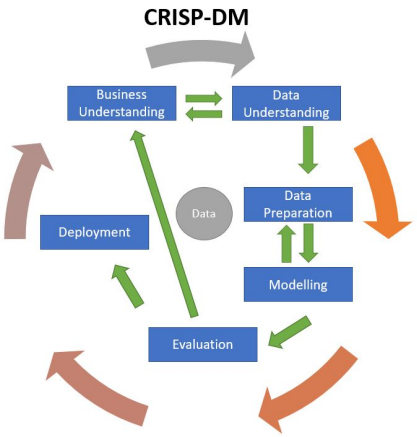


Figure 1.1

1.4 Business Understanding

La fase di Business Understanding è la prima fase del modello CRISP-DM e consiste nella raccolta dei requisiti e di definizione degli obiettivi di business che si intende raggiungere. La fase di Business Understanding prevede la definizione dei business success critiria, ovvero i criteri secondo i quali potremmo accertare che il sistema costruito è in linea con gli obiettivi di business. Nel nostro caso, l'obiettivo di business è quello di stimare, a partire da test di laboratorio e fattori come l'età e il sesso di un paziente, se esso potrebbe essere affetto da diabete.

CHAPTER 2

METODOLOGIA PER LA CLASSIFICAZIONE

2.1 Data Understanding

2.1.1 Acquisizione dei dati

Il dataset utilizzato per l'addestramento del modello può essere scaricato in formato csv da kaggle. La fonte principale dei dati sono le Electronic Health Records (EHRs), ossia delle versioni digitali delle cartelle cliniche dei pazienti che contengono tutta la storia clinica del paziente.

2.1.2 Esplorazione dei dati

I dati si presentano in forma tabellare per un totale di 9 colonne e 94.000 righe. Le 9 caratteristiche del dataset sono:

1. age: Rappresenta l'età del paziente al momento della raccolta dei dati. L'età può fornire indicazioni sul rischio di sviluppare il diabete, poiché l'incidenza della malattia aumenta con l'avanzare dell'età.

2. gender: Può assumere valori 'male' o 'female'. I risultati di alcuni studi hanno determinato che le donne affette da diabete nella gravidanza, avranno una maggior probabilità di contrarre il diabete con l'avanzare dell'età.

3. hypertension: Indica la presenza o l'assenza di ipertensione arteriosa nel paziente. L'ipertensione aumenta il rischio di contrarre il diabete e viceversa, poichè le due condizioni condividono diversi fattori di rischio.

4. heart_diseases: Indica se il paziente ha una storia di malattie cardiache. Le malattie cardiache sono spesso correlate al diabete e possono influenzare il rischio di sviluppare complicanze cardiovascolari.

5. smoking_history: Indica se il paziente ha una storia di fumo. Il fumo può aumentare il rischio di sviluppare il diabete e le sue complicanze.

6. bmi: Il BMI è calcolato dividendo il peso del paziente per il quadrato della sua altezza ed è un indicatore comune dell'obesità. L'obesità è un fattore di rischio significativo per lo sviluppo del diabete.

7. HbA1c_level: L'HbA1c è un indicatore del controllo glicemico a lungo termine e viene utilizzato per monitorare il diabete.

8. blood_glucose_level: Rappresentano i livelli di glucosio nel sangue del paziente. I livelli elevati di glucosio nel sangue sono caratteristici del diabete.

9. diabetes: Indica se il paziente ha avuto il diabete oppure no. Il valore del campo 'diabetes' può valere 0 o 1. Se il valore è 1 significa che il paziente soffre di diabete, 0 se non ha il diabete.

2.1.3 Qualità dei dati

Abbiamo condotto un'analisi sui dati per determinare se fossero presenti dati mancanti. Il risultato di questa analisi è il seguente:

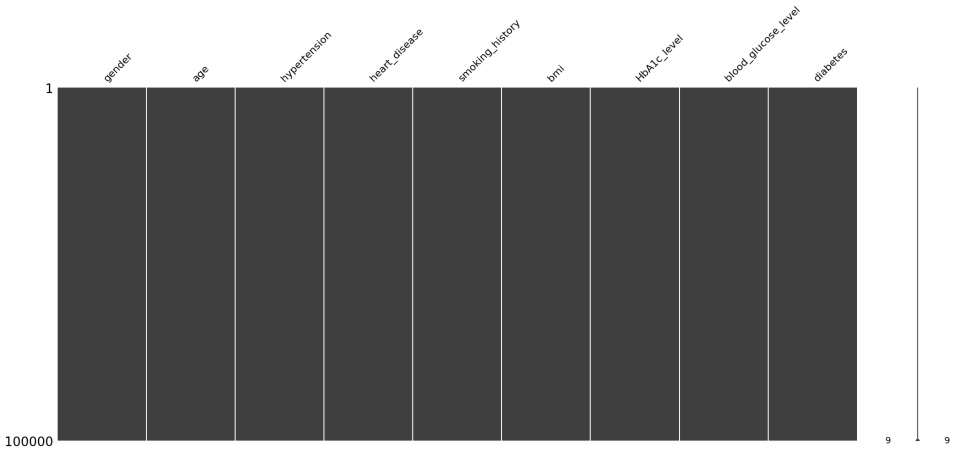


Figure 2.1

Come emerge dal grafico non sono presenti dati mancanti, altrimenti ci sarebbero stati dei punti bianchi in corrispondenza della riga.

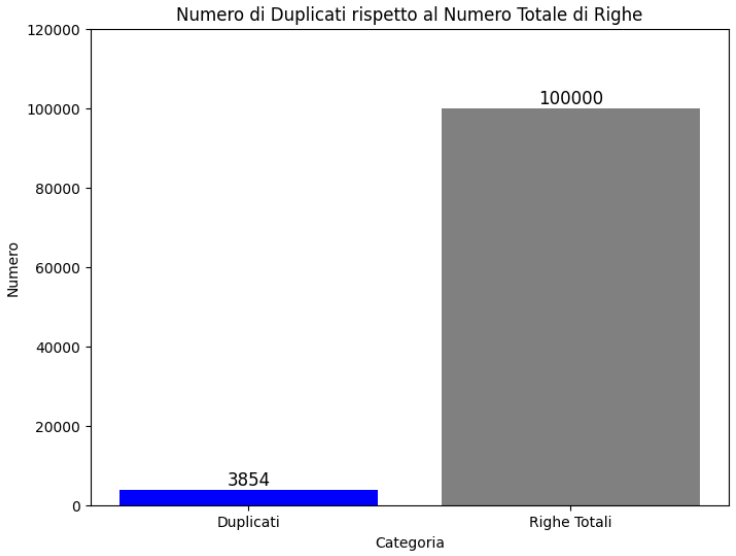


Figure 2.2

Abbiamo trovato la presenza di dati duplicati all'interno del dataset. Secondo una nostra personale riflessione, siamo giunti alla conclusione che la presenza di dati duplicati indica che questi dati hanno una maggiore rilevanza al fine di diagnosticare il diabete.

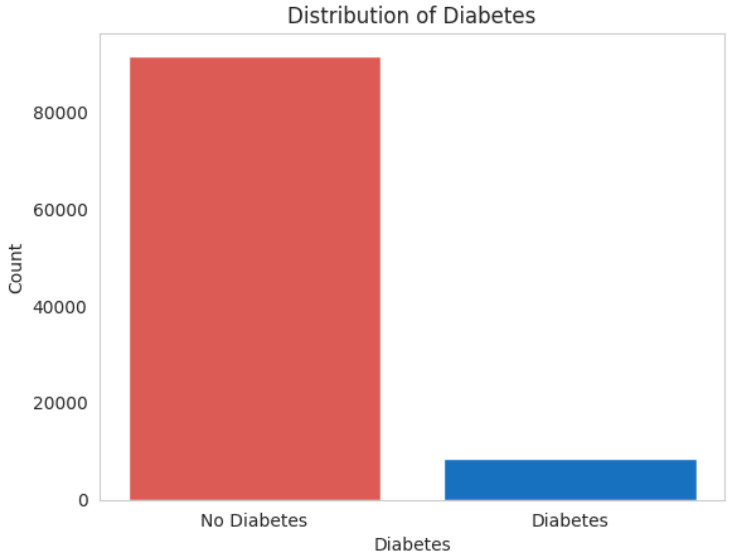


Figure 2.3

Il dataset appare sbilanciato, in quanto la presenza dei diabetici è solo del 9% rispetto ai non diabetici che rappresentano la maggioranza (91%).

2.2 Data Preparation

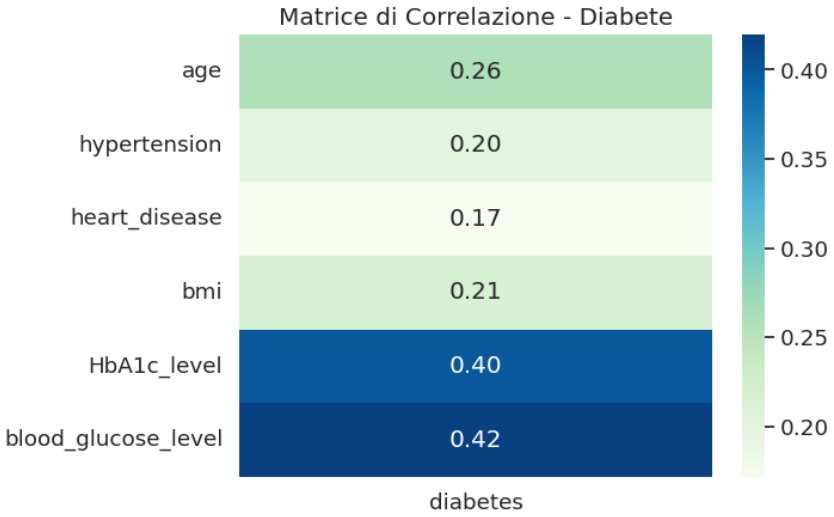
2.2.1 Data cleaning

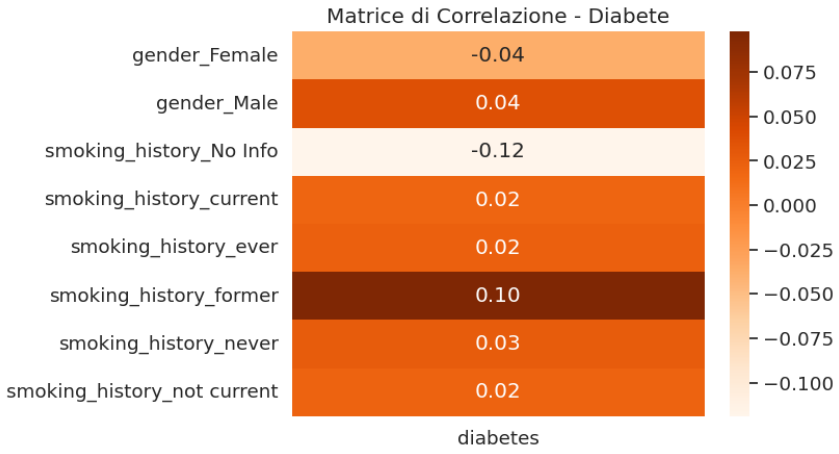
Poichè il dataset non presenta istanze vuote, non è stato necessario eseguire operazioni di data cleaning.

2.2.2 Feature Selection

Durante l'analisi degli attributi abbiamo fatto un'attenta valutazione su ogni singolo attributo, in particolare sulla sua utilità al fine di diagnosticare il diabete. Siccome gli attributi nel dataset sono tutti parametri importanti per diagnosticare il diabete, abbiamo deciso che la migliore soluzione fosse quella di prenderli tutti in considerazione. Ecco, quindi, come compare il dataset alla fine di questa attenta valutazione:

Abbiamo, inoltre, messo in correlazione i vari attributi del dataset con l'attributo "diabetes", al fine di valutare quanto gli attributi si influenzano tra loro. Di seguito riportiamo le **matrici di correlazione**:





Dall'analisi delle matrici di correlazione, è possibile osservare che la presenza di 'HbA1c_level' e 'blood_glucose_level' influenza in modo preponderante la possibilità di contrarre il diabete.

2.2.3 Feature Scaling

Non abbiamo applicato nessun tipo di normalizzazione sui dati che avevamo a disposizione.

2.2.4 Data Balancing

L'oversampling e l'undersampling sono due tecniche utilizzate nell'apprendimento automatico per affrontare il problema dello sbilanciamento di classe in un dataset, in cui una classe è rappresentata da un numero significativamente inferiore di esempi rispetto alle altre classi. Questo sbilanciamento può influire negativamente sulle prestazioni del modello di apprendimento automatico, poiché il modello tende ad essere inclinato verso la classe maggioritaria a causa della sua maggiore presenza di dati di addestramento.
L'oversampling è una tecnica che mira ad aumentare la presenza di esempi della classe minoritaria nel dataset, generando dati sintetici. Uno degli algoritmi di oversampling più utilizzati è l'ADASYN (Adaptive Synthetic Sampling). ADASYN è un metodo basato su campionamento sintetico che mira a generare nuovi esempi sintetici per la classe minoritaria in modo adattivo. L'algoritmo tiene conto della densità dei punti nel dataset e genera più esempi sintetici per le regioni meno dense, consentendo di affrontare meglio le aree più complesse e meno rappresentate della classe minoritaria. In questo modo, ADASYN cerca di bilanciare le distribuzioni delle classi nel dataset.
D'altra parte, l'undersampling è una tecnica che punta a ridurre la presenza di esempi della classe maggioritaria nel dataset, eliminando alcuni dei dati di quella classe. RandomUnderSampler è un metodo di undersampling comune ed efficace che opera selezionando casualmente un sottoinsieme dei dati della classe maggioritaria in modo che il numero di esempi delle classi sia bilanciato. Questo metodo è semplice da implementare e può essere efficace quando il dataset ha una dimensione sufficientemente grande.

In entrambi i casi, è fondamentale valutare attentamente l'impatto delle tecniche di oversampling o undersampling sul modello di apprendimento automatico. È possibile utilizzare misure di valutazione, come l'accuratezza o la precisione per determinare se la tecnica di campionamento ha migliorato le prestazioni del modello rispetto alla situazione di partenza.

CHAPTER 3

MODELING

Terminata la fase di preprocessing, il prossimo step è quello di creare un modello per l'apprendimento

3.1 Gaussian Naive Bayes

Gaussian Naive Bayes è un algoritmo di classificazione ampiamente utilizzato nell'apprendimento automatico. Si basa sul teorema di Bayes e sull'assunzione di indipendenza condizionale delle variabili di input.
L'algoritmo assume che le caratteristiche di input siano distribuite secondo una distribuzione gaussiana (normale).
L'idea alla base di Gaussian Naive Bayes è quella di calcolare la probabilità a posteriori di una classe dato un vettore di caratteristiche di input utilizzando il teorema di Bayes. Questo viene fatto considerando le probabilità a priori delle classi e le probabilità condizionali delle caratteristiche di input date le classi.
L'assunzione di indipendenza condizionale semplifica il calcolo delle probabilità condizionali, poiché considera che le caratteristiche di input siano indipendenti tra loro dato un certo valore di classe. Sebbene questa

assunzione possa non essere realistica in molti casi, Gaussian Naive Bayes è noto per la sua semplicità e velocità di addestramento e classificazione. L'algoritmo è ampiamente utilizzato in applicazioni come il riconoscimento del linguaggio naturale, la classificazione di documenti e altre situazioni in cui le caratteristiche di input possono essere approssimate da una distribuzione gaussiana. Tuttavia, è importante valutare attentamente le assunzioni dell'algoritmo e considerare se sono adatte al problema specifico prima di applicarlo.

3.2 Decision Tree

Un Decision Tree (albero decisionale) è un algoritmo di apprendimento automatico ampiamente utilizzato per problemi di classificazione e regressione. Prende il nome dalla sua struttura a forma di albero, composta da nodi interni che rappresentano decisioni o attributi e foglie che rappresentano le classi o i valori di output.
L'obiettivo principale di un Decision Tree è quello di suddividere il dataset in modo ricorsivo in base agli attributi più significativi, al fine di ottenere una classificazione o una stima precisa. Inizialmente, il Decision Tree seleziona un attributo che meglio separa le istanze in base alla loro classe o valore di output. Successivamente, l'albero si ramifica in base alle possibili combinazioni di valori dell'attributo selezionato, creando sotto-alberi corrispondenti ai rami.
Durante il processo di addestramento, il Decision Tree continua a dividere i dati in modo ricorsivo fino a quando non viene raggiunta una condizione di arresto, ad esempio quando tutte le istanze in una specifica suddivisione appartengono alla stessa classe o quando viene raggiunto un numero massimo di livelli dell'albero.
Un Decision Tree offre numerosi vantaggi, tra cui l'interpretabilità, in quanto il processo decisionale può essere facilmente visualizzato attraverso il percorso dell'albero, e la capacità di gestire sia dati numerici che categorici.

Inoltre, può gestire dati mancanti o rumorosi e può essere combinato con altre tecniche di ensemble per migliorare le prestazioni.
Tuttavia, i Decision Tree possono essere soggetti all'overfitting, specialmente quando l'albero diventa molto profondo e complesso. Per mitigare questo problema, sono state sviluppate varie tecniche come la potatura dell'albero e l'utilizzo di parametri di controllo della complessità.
In conclusione, un Decision Tree è un algoritmo flessibile e intuitivo che può essere utilizzato per la classificazione e la regressione. La sua struttura a forma di albero permette di prendere decisioni basate su attributi significativi, rendendolo uno strumento utile per l'apprendimento automatico e l'analisi dei dati.

3.3 Random Forest

Random Forest è un algoritmo di apprendimento automatico basato su ensemble, che combina l'output di più alberi decisionali per ottenere una previsione più accurata. È ampiamente utilizzato per problemi di classificazione e regressione.
L'idea principale di Random Forest è quella di creare un insieme (ensemble) di alberi decisionali, ognuno dei quali viene addestrato su un sottoinsieme casuale e indipendente del dataset di addestramento. Durante la fase di addestramento, per ogni albero viene selezionato casualmente un sottoinsieme di dati e un sottoinsieme di attributi. Ciò introduce una variazione e una diversità nell'addestramento dei singoli alberi.
Durante la fase di previsione, ciascun albero produce una previsione e la classe (o il valore) che ottiene la maggioranza delle previsioni degli alberi viene considerata come l'output finale del modello di Random Forest.
Random Forest offre diversi vantaggi, tra cui la capacità di gestire grandi dataset con molte caratteristiche, la robustezza agli outlier e la riduzione dell'overfitting rispetto a un singolo albero decisionale. Inoltre, può gestire sia dati numerici che categorici senza la necessità di normalizzazione o

codifica dei dati.
Un altro aspetto interessante di Random Forest è la possibilità di ottenere una stima dell'importanza delle diverse caratteristiche utilizzate per la previsione. Questa informazione può essere utile per identificare le caratteristiche più influenti e comprenderne il ruolo nel processo decisionale. In conclusione, Random Forest è un algoritmo di ensemble che sfrutta la diversità degli alberi decisionali per ottenere previsioni più accurate e stabili. La sua flessibilità, scalabilità e capacità di gestire diversi tipi di dati lo rendono una scelta popolare in diversi contesti di apprendimento automatico.

3.4 Ottimizzazione degli Iperparametri

L'ottimizzazione degli iperparametri è un processo utilizzato nell'ambito del Machine Learning per trovare la combinazione ottimale di iperparametri di un modello. Gli iperparametri sono, di fatto, dei parametri che non vengono appresi automaticamente dal modello di Machine Learning ma devono essere settati manualmente dal programmatore. L'ottimizzazione degli iperparametri può avvenire tramite diverse tecniche. La tecnica utilizzata da noi è la grid search cv. La grid search cv è una tecnica che utilizza la convalida incrociata per valutare le prestazioni di ciascuna combinazione di iperparametri. Quindi, invece di valutare il modello solo su un singolo set di validazione, valuta le prestazioni su più sottoinsiemi dei dati di addestramento. Il risultato del grid search cv ha evidenziato i seguenti valori come iperparametri migliori:

1. max_depth: None
2. min_samples_leaf: 1
3. min_samples_split: 2
4. n_estimators: 50

max_depth: Questo parametro specifica la profondità massima dell'albero di decisione all'interno del Random Forest. Se il valore è impostato su None, l'albero crescerà fino a quando tutte le foglie saranno pure o fino a quando il numero minimo di campioni richiesti per suddividere un nodo non verrà raggiunto. L'uso di un valore limitato per max_depth può aiutare a prevenire l'overfitting, controllando la complessità dell'albero.

min_samples_leaf: Questo parametro specifica il numero minimo di campioni richiesti in una foglia dell'albero di decisione. Se il numero di campioni in una foglia è inferiore a min_samples_leaf, l'albero non eseguirà ulteriori suddivisioni e la foglia sarà considerata pura. Aumentare il valore di min_samples_leaf può contribuire a evitare l'overfitting, consentendo al modello di generalizzare meglio i dati. tramite questo processo siamo riusciti ad avere un miglioramento di 0.01 per quanto riguarda la recall.

min_samples_split: Questo parametro specifica il numero minimo di campioni richiesti per eseguire una suddivisione in un nodo interno dell'albero di decisione. Se il numero di campioni in un nodo è inferiore a min_samples_split, non verranno effettuate ulteriori suddivisioni e il nodo diventerà una foglia. Regolando il valore di min_samples_split, è possibile controllare la complessità dell'albero e limitare l'overfitting.

n_estimators: Questo parametro specifica il numero di alberi decisionali che compongono il Random Forest. Ogni albero viene addestrato su un sottoinsieme casuale dei dati. Aumentando il numero di alberi (n_estimators), il modello avrà una maggiore capacità di apprendimento e potrebbe ottenere prestazioni migliori. Tuttavia, un valore troppo alto di n_estimators può portare a un aumento del tempo di addestramento e della complessità del modello senza migliorare significativamente le prestazioni.

3.5 Validazione

Chiaramente, non si può valutare un classificatore su dati che non si conoscono, altrimenti non si potrebbero misurare le sue prestazioni, ma, avendo un dataset di partenza etichettato, si può sfruttare questa conoscenza per capire come un classificatore classifica questi dati e di conseguenza poter misurare le sue prestazioni. Ma nello stesso tempo, se si addestrasse e si validasse il modello sullo stesso dataset si avrebbero risultati totalmente inaffidabili. Si può però dividere il dataset di partenza in maniera tale da considerare alcune delle istanze come non note. Si creano quindi due insiemi:

1. Il training set, che sarà composto da istanze che l'algoritmo utilizzerà per l'addestramento
2. Il test set, che sarà composto dalle istanze per cui l'algoritmo addestrato dovrà predire la classe di appartenenza e sul quale verrà valutato la sua performance.

La tecnica adottata è stata la K-fold cross validation, la quale consente di dividere il dataset in k parti e a ogni iterazione, di usare 1 parte per la validazione e le k-1 parti restanti per l'addestramento; ovviamente, a termine della validazione, il modello viene eliminato. Al parametro k è stato assegnato valore 10, mentre la k fold è stata ripetuta 50 volte al fine di minimizzare il Mean Absolute Error. Abbiamo implementato tale algoritmo tramite la libreria sklearn che mette a disposizione il modulo KFold.

CHAPTER 4

ANALISI DEI RISULTATI

4.1 Valutazione

A prescindere dalla procedura di validazione che si utilizzerà, si avrà bisogno di strumenti adatti per valutare la bontà delle predizioni. Per avere una rappresentazione dell'accuratezza dei 3 classificatori si fa impiego della matrice di confusione. La matrice di confusione è una matrice con cui poter indicare se ed in quanti casi il classificatore ha predetto correttamente o meno il valore di un'etichetta del test set.

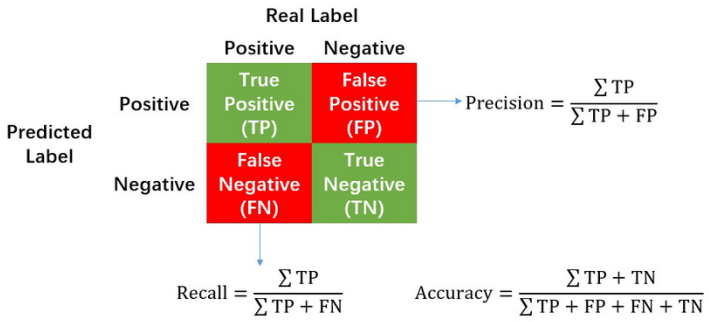


Figure 4.1

Sulla base dei valori della matrice di confusione, si procede a calcolare diverse metriche di valutazione; le principali sono Precision, Recall,Accuracy ed

F-score

La **precision** indica il numero di predizioni corrette rispetto tutte le predizioni fatte dal classificatore. In altri termini, indica il numero di errori che ci saranno nella lista delle predizioni fatte dal classificatore.

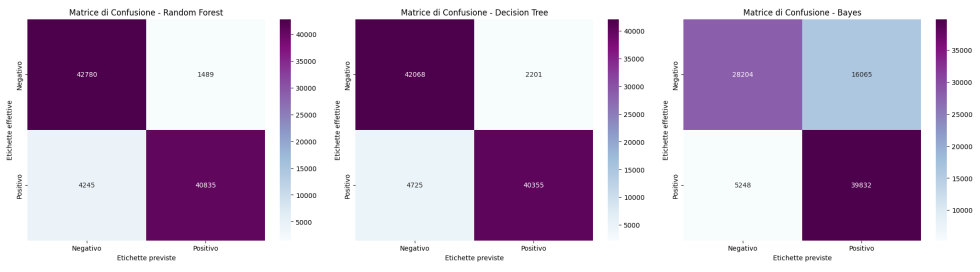
La **recall** indica il numero di predizioni corrette rispetto a tutte le istanze positive di quella classe. In altri termini, indica quante istanze positive nell'intero dataset il classificatore può determinare.

L'accuracy indica il numero totale di predizioni corrette, considerando anche i veri negativi al numeratore. Questo potrebbe creare un problema di interpretazione nel caso di dataset sbilanciati dove il numero di casi positivi è molto basso.

F-score rappresenta la media ponderata della precision e della recall.

4.2 Valutazione dei 3 modelli

Sono state calcolate le matrice di confusione e le quattro metriche di valutazione sui 3 modelli per determinare e scegliere il migliore in termini di performance. Questo perchè per lo stesso problema ci sono più algoritmi possibili da poter utilizzare ma solo uno deve essere scelto. Ci si affida al metodo empirico, misurando l'errore commesso da ciascun algoritmo. Il metodo empirico e la misura dell'errore offrono una base di confronto tra i vari modelli, andando ad eseguire gli algoritmi nel contesto del problema, e misurando le loro prestazioni in base agli errori commessi.



Dalle matrice di confusione sono state calcolate le quattro metriche di valutazione.

```
NAIVE BAYES - Metriche di valutazione:
Accuracy: 0.76
Precision: 0.71
Recall: 0.88
F1-Score: 0.78
```

```
DECISION TREE - Metriche di valutazione:
Accuracy: 0.92
Precision: 0.95
Recall: 0.90
F1-Score: 0.91
```

```
RANDOM FOREST CLASSIFIER - Metriche di valutazione:
Accuracy: 0.94
Precision: 0.97
Recall: 0.91
F1-Score: 0.93
```

Così come si evince dalle metriche di valutazione, l'algoritmo migliore è il Random Forest, con un f1-score di 0.93, che ha portato alla sua scelta nonostante presenti degli svantaggi rappresentati nella seguente tabella:

Classificatore	Vantaggi	Svantaggi
Gaussian Naive Bayes	Naive Bayes è un algoritmo semplice e intuitivo da implementare. Ha un'elevata efficienza computazionale, infatti è molto veloce nella sua esecuzione	È il peggiore in quanto a performance, poichè in fase di predizione considera tutte le caratteristiche indipendenti tra di loro
Decision Tree	Ha performance molto simili al Random Forest. La costruzione e l'utilizzo dei Decision Tree sono relativamente semplici.	I Decision Tree tendono ad adattarsi troppo ai dati di addestramento. Ciò può portare a un overfitting, in cui il modello è troppo specifico per il set di addestramento e generalizza male su nuovi dati.
Random Forest	Riesce a gestire in modo efficiente dataset con grandi quantità di dati. È il migliore tra i tre, raggiungendo un'accuracy e una precision molto alta	È il più costoso e lento da realizzare andando a creare un insieme di alberi decisionali

CHAPTER 5

CONCLUSIONI

In questo progetto sono stati utilizzati metodi di apprendimento supervisionato su un dataset. In una prima parte ci si è occupati di tutte le fasi che comprendono la preparazione dei dati affinchè possano essere utilizzati dai classificatori. Nella seconda parte si è fatto un focus sulle varie tipologie di algoritmi di classificazione, sperimentando empiricamente allo scopo di determinare il migliore. È emerso che il Random Forest è l'algoritmo migliore in questo caso. Nonostante gli ottimi risultati emerge comunque la necessità di migliorare il modello utilizzando dataset più precisi e vasti.

5.1 Repository Github

Link al progetto