UNIVERSITÁ DEGLI STUDI DI MILANO-BICOCCA

Corso di Machine Learning

Lorenzo Megna — Matr. 868929

Lorenzo Molinari — Matr. 911610

Massimo Trippetta — Matr. 869286

Anno Accademico 2023/24

Indice

Introduzione

0.1 Obiettivo

Lo scopo della costruzione del modello oggetto di questo elaborato è la creazione di un classificatore, che basandosi sulle informazioni fornite dalle diverse variabili, mira a classificare correttamente gli individui come diabetici (1) o non diabetici (0).

0.2 Dataset

Abbiamo selezionato un dataset focalizzato su informazioni cliniche riguardanti i casi diabete. Il dataset è composto da diverse variabili chiave, ognuna delle quali contribuisce a delineare un quadro completo della condizione diabetica. Di seguito, il dettaglio delle variabili considerate:

1. **Genere (Gender)**: Questa variabile rappresenta il sesso biologico dell'individuo, un fattore che può influenzare significativamente la suscettibilità al diabete.

2. **Età (Age)**: L'età è un elemento cruciale poiché il diabete è più comunemente diagnosticato negli anziani. Nel nostro insieme di dati, l'età varia da 0 a 80 anni, coprendo così un ampio spettro di fasce di età.

3. **Ipertensione (Hypertension)**: Rappresenta la presenza o assenza di ipertensione, una condizione medica caratterizzata da pressione sanguigna costantemente elevata. La variabile assume valori 0 o 1, dove 0 indica l'assenza di ipertensione e 1 indica la presenza di questa condizione.

4. **Malattie Cardiache (Heart Disease)**: Questa variabile binaria indica se l'individuo presenta o meno malattie cardiache, un'altra condizione medica correlata all'incremento del rischio di sviluppare il diabete.

5. **Abitudini del fumo (Smoking History)**: L'abitudine al fumo è considerata un fattore di rischio per il diabete. Nel nostro dataset, questa variabile presenta cinque categorie: "not current," "former," "No Info," "current," "never," e "ever".

6. **Indice di Massa Corporea (BMI)**: L'indice di massa corporea è una misura del grasso corporeo basata sul peso e sull'altezza. Valori più elevati di BMI sono associati a un maggiore rischio di diabete. Il nostro dataset copre un intervallo di BMI da 10,16 a 71,55.

7. **Livello di HbA1c (HbA1c Level)**: Questo rappresenta il livello medio di zucchero nel sangue di un individuo negli ultimi 2-3 mesi. Livelli più alti indicano un rischio maggiore di sviluppare il diabete, con un valore di HbA1c superiore al 6,5% che indica la presenza di diabete.

8. **Livello di Glucosio nel Sangue (Blood Glucose Level)**: Questa variabile riflette la quantità di glucosio nel flusso sanguigno in un dato momento. Livelli elevati di glucosio sono un indicatore chiave del diabete.

9. **Diabete (Diabetes)**: Questa è la variabile target prevista, con valori pari a 1 che indicano la presenza di diabete e 0 che indicano l'assenza di diabete.

0.3 Librerie python

Per la costruzione del classificatore abbiamo utilizzato diverse librerie python:

1. **Pandas**: Utile per la manipolazione e analisi dei dati (strutture dati), viene utilizzata nel progetto per la preparazione, esplorazione e pulizia dei dati.

2. **Numpy**: Libreria inerenti il calcolo scientifico, permette operazioni efficienti su dati multidimensionali ed equazioni matematiche veloci.

3. **Matplotlib**: Permette la visualizzazione dei dati tramite grafici, permette inoltre di poter visualizzare graficamente i risultati dei vari modelli provati.

4. **Sklearn**: Scikit-Learn, libreria che permette di implementare vari algoritmi di apprendimento automatico, rende possibile la selezione di modelli, valutazione delle prestazioni e l'eventuale pre-elaborazione dei dati.

5. **Seaborn**: Libreria di visualizzazione basata su Matplot per la creazione di grafici statistici.

6. **Keras**: Libreria che mette a disposizione funzioni per la costruzione, addestramento e sperimentazione delle reti neurali (modelli di deep learning).

7. **TQDM**: Libreria che fornisce una barra di avanzamento in stile console.

8. **SciPy**: Libreria Open-Source che mette a disposizione funzionalità di ottimizzazione per l'algebra lineare e statistica, si prevede un utilizzo in combinazione con numpy per operazioni matematiche avanzate.

9. **Time**: Libreria per la gestione del tempo di un programma Python, ci permette di misurare il tempo di esecuzione.

Capitolo 1

Analisi iniziali e dati di lavoro

1.1 Analisi esplorativa

La prima vera e propria fase di lavoro è stata l'analisi esplorativa, fase finalizzata alla comprensione a fondo del dataset in uso. Questa fase ci permette di fornire insights importanti per la fase di preparazione dei dati e l'implementazione di modelli di machine learning. Per ottenere una comprensione approfondita dei dati, abbiamo utilizzato la funzione data.describe() sulla nostra base di dati, che ci ha fornito una serie di risultati e statistiche riguardo il data frame in esame, tra cui: conteggio, media, dispersione min, max e altro ancora:

	age	bmi	HbA1c_level	blood_glucose_level
count	100000.000000	100000.000000	100000.000000	100000.000000
mean	41.875660	27.320768	5.527507	138.058060
std	22.535417	6.636784	1.070672	40.708136
min	0.000000	10.010000	3.500000	80.000000
25%	24.000000	23.629999	4.800000	100.000000
50%	43.000000	27.320000	5.800000	140.000000
75%	60.000000	29.580000	6.200000	159.000000
max	80.000000	95.690002	9.000000	300.000000

Figura 1.1: Analisi descrittive delle variabili

Importante inoltre visualizzare la distribuzione della variabile "diabete" nel database che come risultato ha evidenziato che la categoria "0" (non diabete) ricopre il 91.52% del totale, mentre la categoria "1" (diabete) ricopre l'8.48% del totale,

questo ci porterà, successivamente, a bilanciare il dataset. Da queste informazioni abbiamo estrapolato il seguente istogramma:

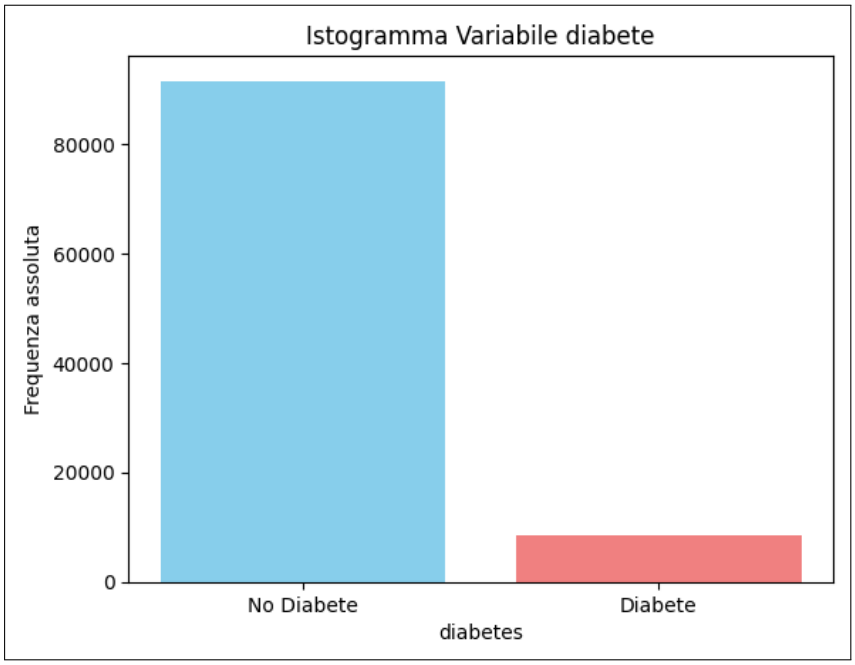

Figura 1.2: Distribuzione della variabile diabete

Nella fase successiva abbiamo esaminato i vari campi delle variabili ed in particolare ci siamo soffermati sull'analisi della variabile smoking history:

smoking_history	diabetes	
No Info	0	34362
	1	1454
current	0	8338
	1	948
ever	0	3532
	1	472
former	0	7762
	1	1590
never	0	31749
	1	3346
not current	0	5757
	1	690

Figura 1.3: Distribuzione variabile smoking history

Da questi dati possiamo notare che smoking history ha 35000 valori identici, tutti pari a “no info”, valori che ci fanno perdere moltissime informazioni, abbiamo preso la decisione di eliminare questa variabile dal dataframe in modo che la scarsità di queste informazioni non influisca negativamente sulla qualità del dataset.
Abbiamo proseguito con l’analisi degli altri dati in particolare focalizzandoci su aspetti che abbiamo ritenuto interessanti, per prima cosa notiamo la distribuzione della variabile “age” in funzione della variabile “diabetes” attraverso un boxplot, il quale ci fa notare come il diabete sia maggiormente presente nelle persone anziane, con una mediana intorno ai 60 anni.

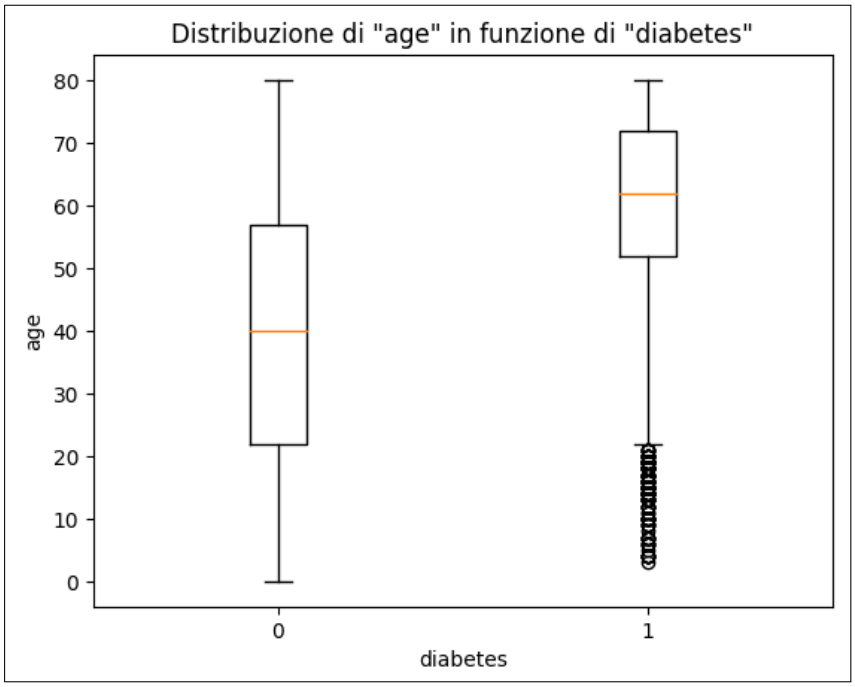

Figura 1.4: Boxplot Age-Diabete

Successivamente attraverso la scatter matrix possiamo verificare la linearità tra le componenti del dataset e la dispersione tra i vari attributi:

Infine, attraverso la heatmap, vediamo come nessuna variabile sia particolarmente correlata con altre variabili, avendo il valore più alto pari a 0.34 tra ’bmi’ e la variabile riguardante l’età.

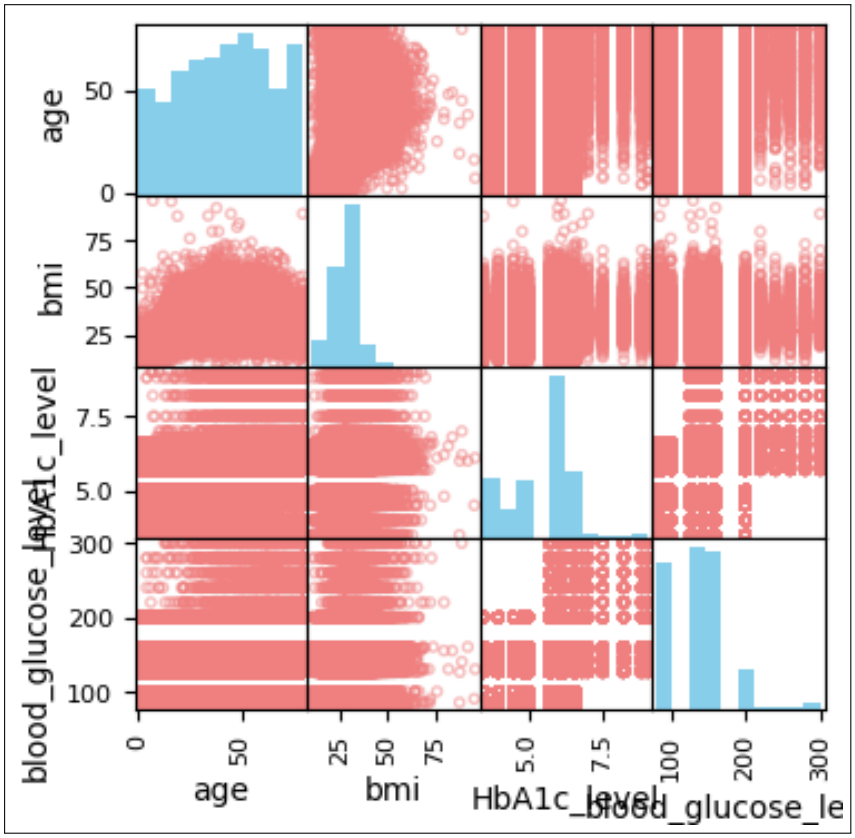

Figura 1.5: Scatter matrix

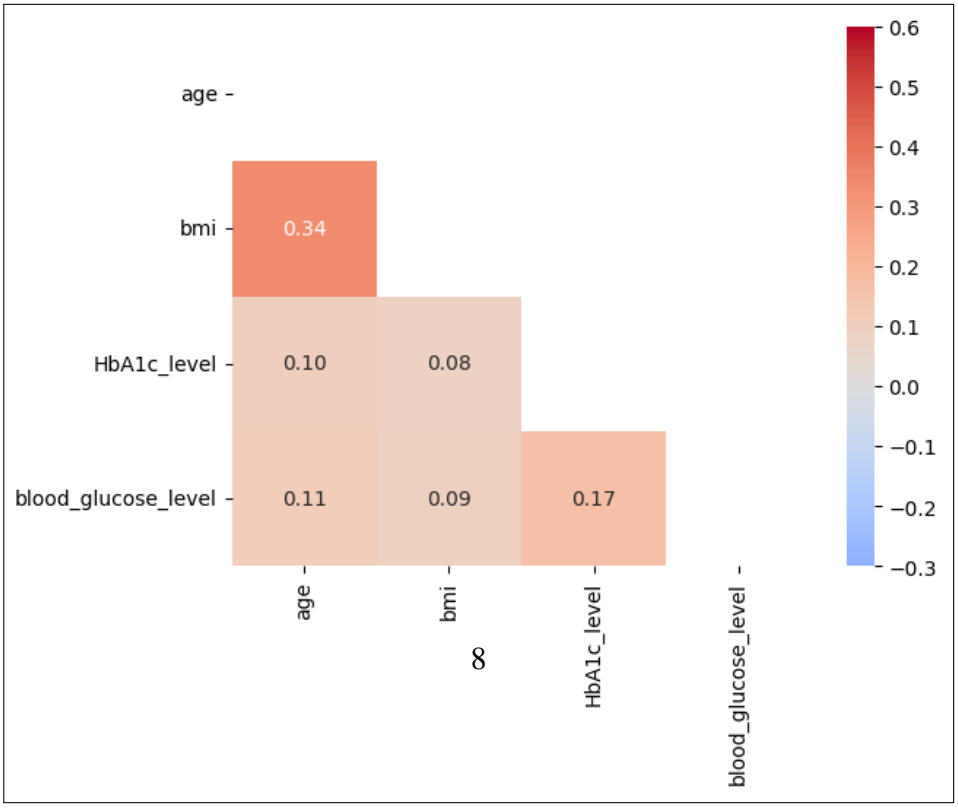

Figura 1.6: Heatmap

1.2 Bilanciamento dei dati

Come evidenziato durante l'analisi esplorativa, il dataframe mostra una netta sproporzione verso i casi di "non diabete". Di conseguenza, abbiamo formulato l'idea di bilanciare la distribuzione delle classi attraverso una strategia combinata di oversampling per i casi di diabete (corrispondenti alla variabile "diabete" con valore 1) e successivo undersampling stratificato per i casi di non diabete (rappresentati dalla variabile "diabete" con valore 0).
In particolare, attraverso l'oversampling vogliamo aumentare il numero di casi di diabete nel dataset, garantendo in questo modo che il modello di machine learning riceva una rappresentazione più bilanciata di entrambe le classi. Successivamente, effettueremo l'undersampling alla classe di non diabete, mantenendo così una distribuzione proporzionale alle classi originarie. In questo modo possiamo diminuire gli effetti della sproporzione iniziale e migliorare le prestazioni dei vari modelli di cui analizzeremo il comportamento.

1. Per prima cosa, abbiamo scelto di generare istanze "artificiali" esclusivamente nel test set. Questa decisione è stata presa con lo scopo di evitare una possibile distorsione nella fase di validazione dei modelli utilizzati. Infatti, mantenendo la creazione di istanze sintetiche limitata nel test set, abbiamo cercato di garantire che la performance del modello sia valutata su un insieme di dati rappresentativo della distribuzione reale, senza essere influenzata dalla presenza di dati sintetici così da preservare l'integrità dei dati.
   In particolare abbiamo suddiviso in: set di addestramento (65% delle istanze) e set di test (35% delle istanze).

2. Siamo poi passati alla fase di Oversampling, dato che la funzione "SMOTE" non accetta parametri rappresentati da valori binari o stringhe, abbiamo optato per la suddivisione del dataset in 3 parti, ovvero una per ciascuna delle nostre variabili binarie.
   Abbiamo così ottenuto il nuovo valore sulla suddivisione del dataset, ovvero la categoria "0" (non diabete) ricopre ora il 66,67% del totale, mentre la categoria "1" (diabete) ricopre il 33,33% del totale. L'aggiunta è stata di circa 28 mila nuove righe con target 1 (che possiamo tenere).

3. La fase successiva riguarda l'Undersampling stratificato sulle istanze con target 0; abbiamo poi proceduto con un'operazione di undersampling, utilizzando il metodo sample con un parametro di frazione pari a 0.65 (65% delle istanze originali), abbiamo ridotto in questo modo il numero di istanze.

4. Ultima fase è stata quella dell'unione delle nuove istanze create attraverso le fasi di Oversampling e Undersampling al fine di riottenere il dataset di test

completo per poi procedere ed effettuare le prime operazioni sui modelli. Il dataset risultante è ora più bilanciato ed è distribuito: al 57% delle variabili corrisponde al valore 0, mentre il 33% corrisponde al valore 1.

Capitolo 2

Albero

2.1 Albero Completo

Il primo modello che abbiamo deciso di implementare per il nostro studio è l'albero decisionale implementato nella libreria Scikit-learn (SKlearn).
In particolare, questa implementazione, richiede che le variabili coinvolte siano di natura numerica, di conseguenza, si è resa necessaria una procedura di conversione delle variabili categoriche in variabili numeriche.
A tal fine, abbiamo utilizzato la classe LabelEncoder, che ha consentito la trasformazione delle variabili categoriche 'gender', 'hypertension', e 'heart disease' in variabili numeriche.

	gender	hypertension	heart_disease	age	bmi	HbA1c_level	blood_glucose_level
80002	0	0	0	36	34	3	160
24711	0	0	0	74	27	5	85
5732	0	0	0	65	27	4	100
72459	0	0	0	17	23	6	159
72384	0	0	0	56	23	4	160

Figura 2.1: Label Encoder

Subito dopo la fase di preparazione dei dati abbiamo proceduto con l'adattamento del modello degli alberi decisionali, ovvero abbiamo eseguito il fitting dell'albero completo. In questa fase, abbiamo utilizzato il DecisionTreeClassifier, con specifica del parametro random state fissato a 42 per garantire la riproducibilità dei risultati.

Tenendo conto che la classificazione binaria corrisponde a:

- C(0,0): Veri negativi
- C(0,1): Falsi positivi
- C(1,0): Falsi negativi
- C(1,1): Veri positivi

Abbiamo eseguito la predizione dell'albero completo ottenendo i seguenti risultati:

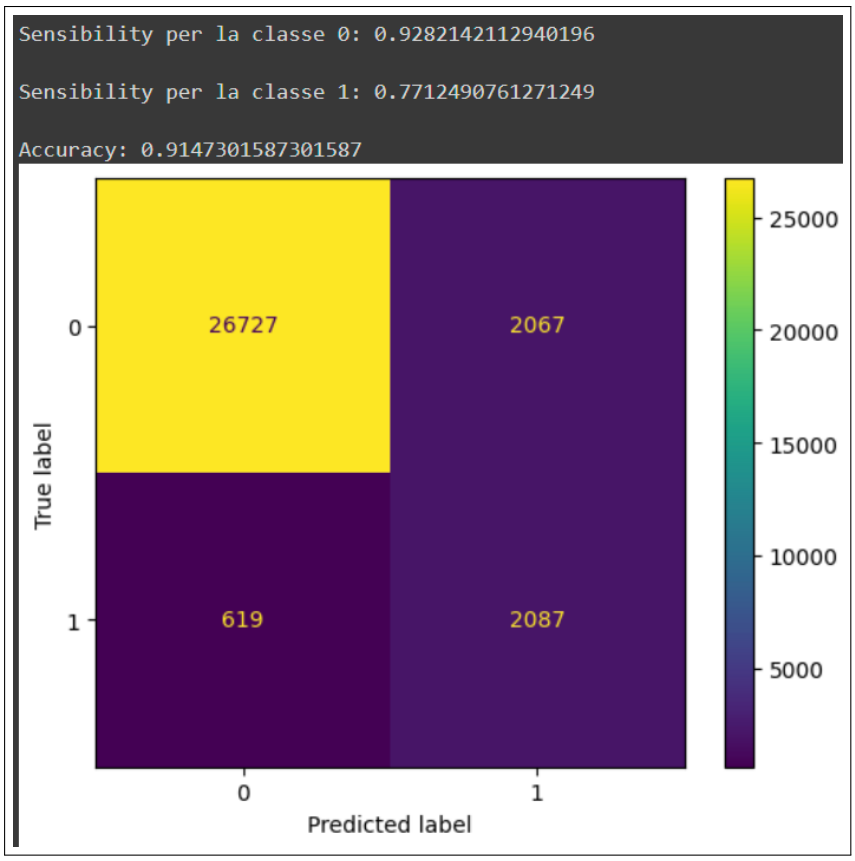

Figura 2.2: Matrice di confusione full tree

2.2 Param Grid

Successivamente, abbiamo proceduto all'implementazione della tecnica di pruning dell'albero mediante l'utilizzo della Cross-Validated Cost-Complexity Pruning (CCP Alpha). Metodologia di pruning che si basa sull'idea di consentire

all'albero di crescere fino a raggiungere la massima espansione e successivamente "potarlo" per ottenere una versione più compatta ed efficiente. L'obiettivo di questo processo è trovare un equilibrio tra la complessità dell'albero e la minimizzazione degli errori di classificazione.
La nostra strategia è stata incentrata sul bilanciamento della gestione della complessità dell'albero e dell'errore di classificazione. Un albero eccessivamente grande rischia di sovrastimare i dati (overfitting), mentre un albero con pochi nodi potrebbe sottostimare i dati (underfitting).
Durante l'esecuzione di questo algoritmo, abbiamo registrato i risultati in vettori, consentendoci di visualizzare le variazioni delle metriche di valutazione in relazione alla complessità dell'albero. L'obiettivo finale è stato quello di individuare il valore ottimale di CCP Alpha, che rappresenta il punto di compromesso ideale tra la complessità dell'albero e le performance predittive.
Un ulteriore passo significativo è stato rappresentato dalla fase di Grid-search, procedura attraverso la quale abbiamo esplorato delle possibili configurazioni dei parametri per il nostro albero decisionale al fine di identificare la combinazione ottimale. Precisazione importante è che questa analisi è stata condotta localmente su una macchina locale per ottimizzare i tempi di esecuzione.
Il risultato che abbiamo ottenuto dall'esecuzione della Grid-search è stato:

	split_criterior	split_stategy	max_depth	ccp_alphas	precision %	accuracy
0	gini	best	3	0.000000	100.0	96.663492
9155	entropy	best	4	0.000002	100.0	96.663492
9157	entropy	best	4	0.000002	100.0	96.663492

Figura 2.3: Risultati Grid-search Albero

- Come criterio per valutare la qualità è risultato "gini" che indica che l'algoritmo sta massimizzando l'informazione ottenuta da ogni suddivisione cercando di mantenere purezza delle classi nei nodi dell'albero.
- Come metodo di splitting è risultato "best", miglior metodo per la suddivisione dell'albero in base al criterio ottenuto.
- Come massima profondità "3", in modo da evitare eventuali overfitting e underfitting
- Come CCPv Alpha risultato è stato "0"

2.3 Albero finale

Nella fase finale del processo di modellazione dell'albero decisionale ci siamo dedicati all'addestramento del modello mediante l'applicazione della configurazione ottimale identificata nei parametri precedenti. Dopodiché abbiamo valutato le performance del modello addestrato mediante il suo impiego sul set di dati di test (final test set). In modo da testare l'efficacia del modello su dati pregressi e verificare la sua capacità di generalizzazione.

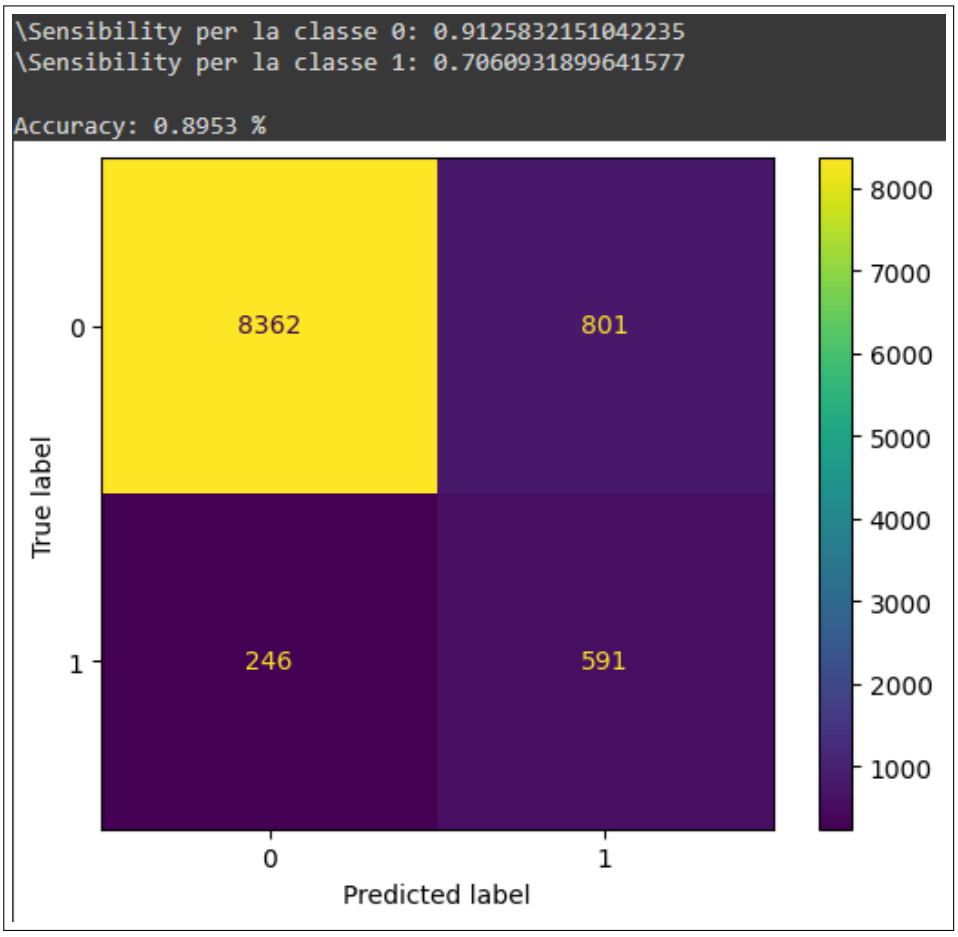

Figura 2.4: Matrice di confusione dell'albero con la migliore configurazione di parametri

Infine possiamo anche visualizzare la distribuzione dell'albero:

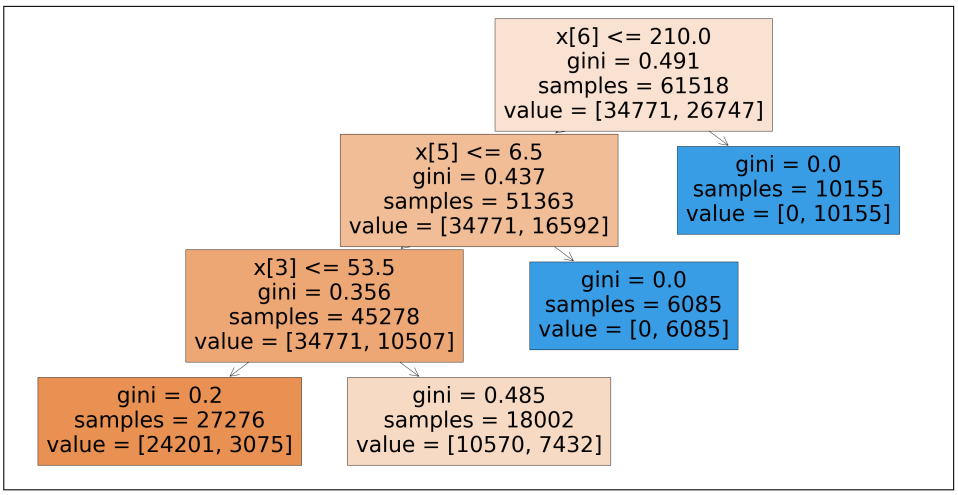

Figura 2.5: Visualizzazione distribuzione Albero finale

Capitolo 3

Rete Neurale

3.1 Preparazione dati

La prima fase per lo sviluppo di un modello di machine learning basato sulle reti neurali ha richiesto la conversione delle variabili categoriche, ovvero 'gender', 'hypertension', e 'heart disease', in una codifica numerica utilizzando il LabelEncoder. Trasformazione richiesta in quanto per lavorare con questo tipo di reti neurali vengono richiesti input di carattere numerico anziché categorici.

3.2 Scelta layers

Il successivo passo nell'elaborazione del modello a rete neurale ha richiesto la determinazione del numero di strati (layers) da impiegare all'interno del nostro progetto, ci siamo ispirati a questo articolo[1] che ci ha fornito un aiuto per la determinazione del numero di neuroni nascosti nell'architettura della rete neurale.

Abbiamo quindi sperimentato diverse combinazioni di neuroni per il modello fino ad identificare l'architettura che ha prodotto i risultati più soddisfacenti. La configurazione finale del modello prevede quindi:

- Un primo strato di input con 7 neuroni, corrispondenti alle 7 features in input, con una funzione di attivazione lineare, ideale per il layer di input.
- Il secondo strato comprende 5 neuroni, poiché la grandezza del layer centrale deve essere compresa tra la dimensione del layer di input e quella del

[1] https://medium.com/geekculture/introduction-to-neural-network-2f8b8221fbd3#: :text=The%20number%20of%20hidden%20neurons,size%20of%20the%20input%20layer

layer di output, in generale, il numero di neuroni nascosti deve essere i due terzi della grandezza dell'input layer, sommato alla grandezza del layer di output. In questo caso utilizziamo una funzione di attivazione "tanh".

- Il terzo strato contiene 4 layer, attivati da una funzione ReLu.
- L'ultimo strato è formato da 2 neuroni, riflettendo il fatto che l'output è rappresentato da un vettore di due componenti (0,1) con la funzione sigmoide.
- Il numero di epoche è di 20 poichè dopo la 15-esima epoca (circa) l'accuratezza e il valore della loss non miglioravano in modo importante. Abbiamo quindi deciso di tenere un numero esiguo di epoche.

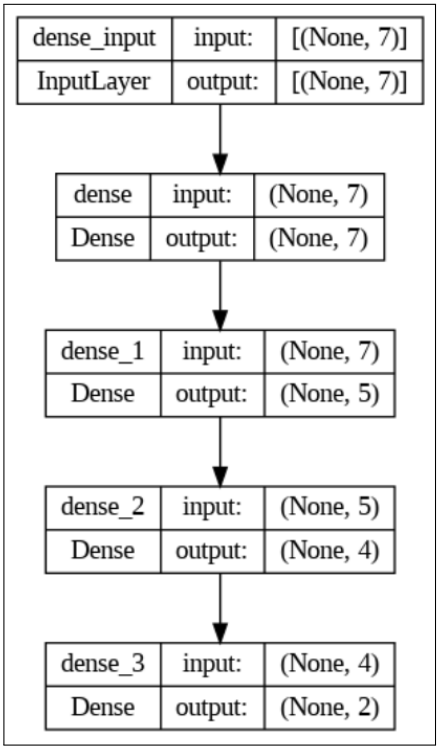

Figura 3.1: Struttura della rete neurale

3.3 Rete neurale finale

Passo successivo è stata la valutazione del modello utilizzando il set di test che ci ha dato i seguenti risultati:

- test loss = 0.222 (quantità di errore prodotta dal modello)
- Sensibilità: 91% (tasso di veri positivi)
- Accuratezza: 85% (rapporto tra predizioni corrette e il numero totale di predizioni)

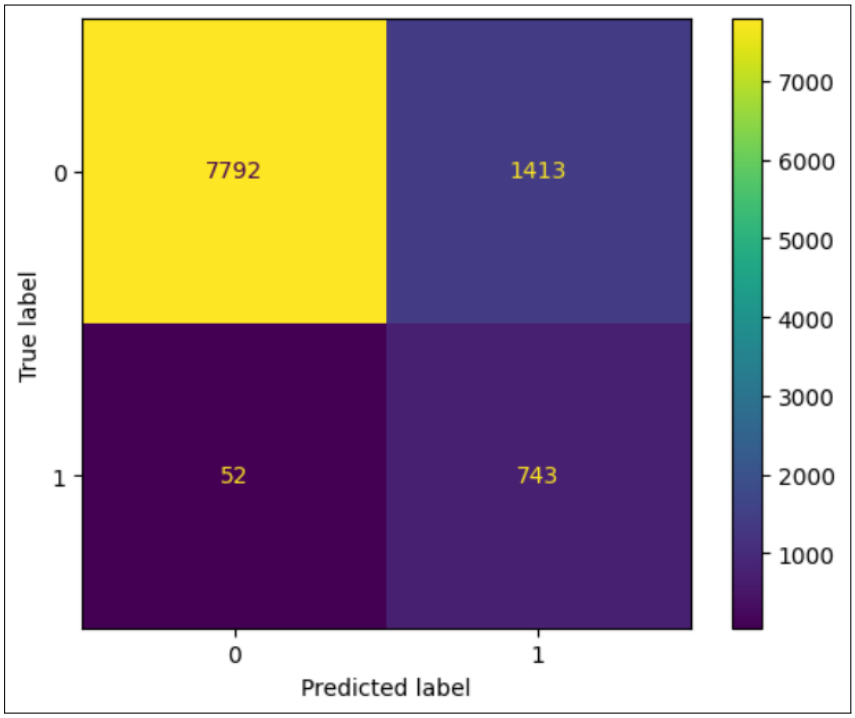

Figura 3.2: Matrice di confusione della rete neurale

Capitolo 4

Support Vector Machine

4.1 Preparazione dati

L'ultimo modello che abbiamo implementato è un SVM [Support Vector Machine], come per gli altri modelli prima di poterlo correttamente implementare è fondamentale preparare il nostro dataset in modo che sia correttamente ottimizzato e funzioni alla perfezione.
Infatti, come per gli altri modelli, utilizziamo l'encoder per trasformare le variabili categoriali i numeriche e successivamente effettuiamo anche una trasformazione degli array multidimensionali associati alle etichette in array monodimensionali attraverso l'utilizzo della funzione:

```
ravel()
```

in NumPy che consente di appiattire la struttura originale degli array, convertendoli tutti in un vettore monodimensionale, questa operazione è utile in quanto semplifica la rappresentazione delle etichette, rendendole più adatte per l'utilizzo degli algoritmi SVM.
Dopo aver finito di preparare il dataset siamo passati all'inizializzazione del modello SVM, impostando come parametro C o anche detta variabile di slack (termine di regolarizzazione, ovvero utile per il controllo e la penalizzazione per gli errori di classificazione) 1 ed il parametro kernel il kernel radiale (RBF)utile per l'utilizzo nei modelli non lineari. Ulteriore fase è stata l'addestramento del modello, in modo che SVM trovi il margine ottimale tra le classi di input e quindi trovare l'iperpiano ottimale che separi nel modo migliore le classi.

4.2 SVM iniziale

Abbiamo poi valutato il modello:

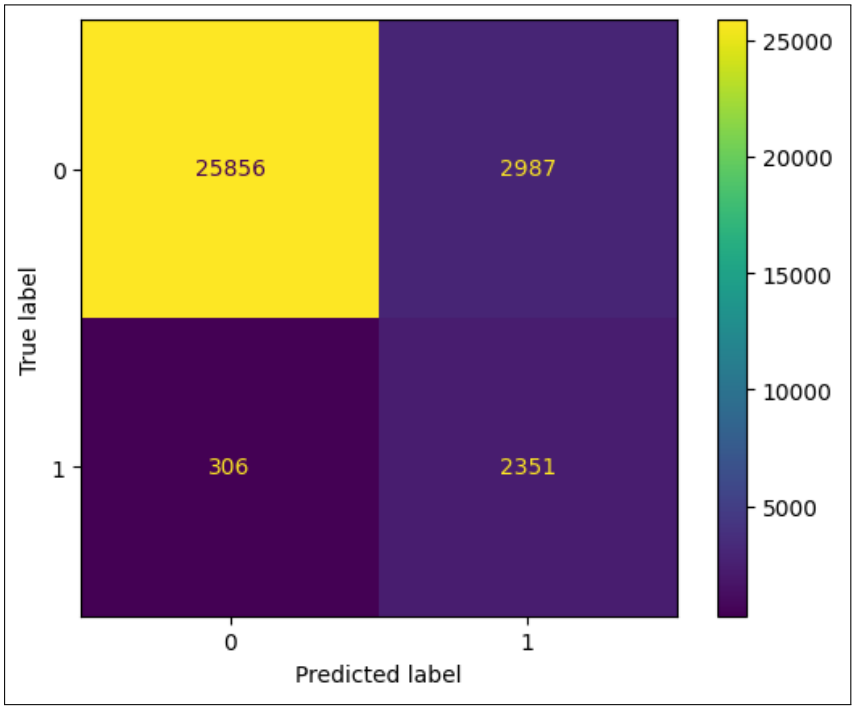

Figura 4.1: Matrice di confusione della SVM iniziale

Il modello iniziale ci offre una accuracy pari a 0.895, una sensibility per la classe 0 pari a 0.896 e per la classe 1 uguale a 0.884, riuscendo quindi a classificare molto bene i malati di diabete.

4.3 Param Grid

Fase fondamentale anche per il modello SVM è il param-grid, ovvero la ricerca dei parametri ottimali per la separazione tramite iperpiano, che ha dato come esito: [9801, 'rbf'] ovvero come parametro C: 9801 (ovvero la variabile Slack) e come kernel: RBF.

variabile slack	kernel	Sensibility %	accuracy
9801	rbf	80.216216	87.652745
9601	rbf	80.216216	87.616268
9201	rbf	80.162162	87.598030

Figura 4.2: Risultati paramgrid SVM

4.4 SVM finale

Ottenuta la migliore configurazione di parametri, siamo poi andati a fittare il modello SVM finale con i parametri ottenuti dalla param grid:

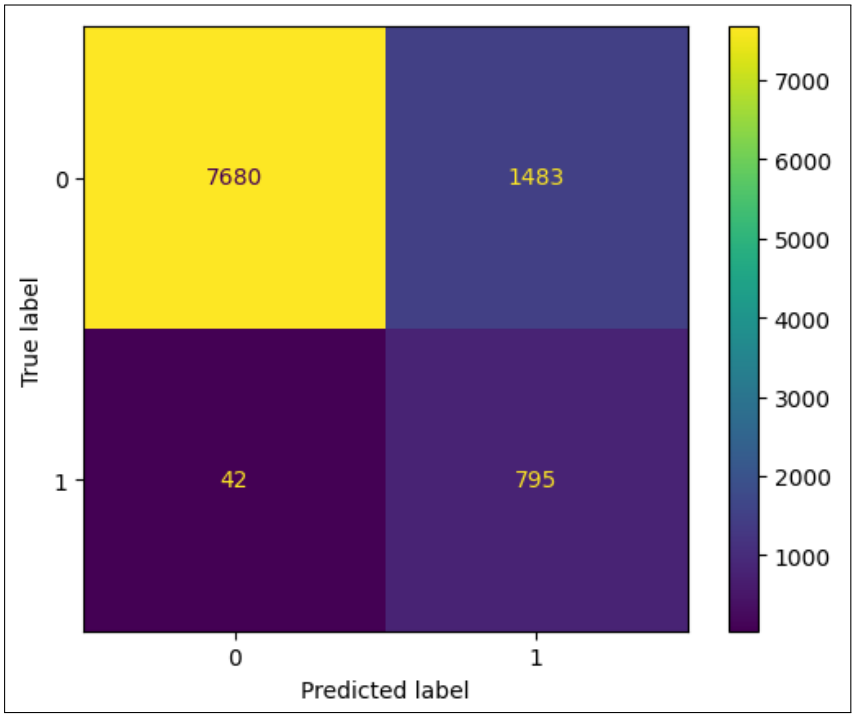

Figura 4.3: Matrice di confusione della SVM finale

Il modello finale testato sul validation set ha una sensibility molo alta (0.949) e anche un ottima accuracy (0.847).

Conclusioni

Per la parte finale del nostro elaborato andiamo a valutare e confrontare i modelli attraverso la curva ROC (Receiver Operating Characteristic) e l'area sotto la curva (AUC), una tecnica comunemente utilizzata per valutare le prestazioni dei modelli di classificazione.
La curva ROC si calcola tracciando il tasso di vero positivo (True Positive Rate, TPR) rispetto al tasso di falso positivo (False Positive Rate, FPR) al variare della soglia di decisione del modello. L'AUC rappresenta l'area sotto questa curva ROC.
Una curva ROC ideale si avvicina rapidamente all'angolo superiore sinistro del grafico, il che indica un alto TPR e un basso FPR. Un AUC di 1.0 indica un modello perfetto, mentre 0.5 indica un modello che non è migliore di una scelta casuale.
Nel nostro caso il modello che, inizialmente si avvicina di più all'angolo sinistro è l'albero, ma risulta avere un TPR non troppo alto, quindi, in questo caso, il modello migliore è l'SVM, il quale si avvicina ad un TPR pari ad 1 e risulta anche essere il modello con l'AUC maggiore(0.89).

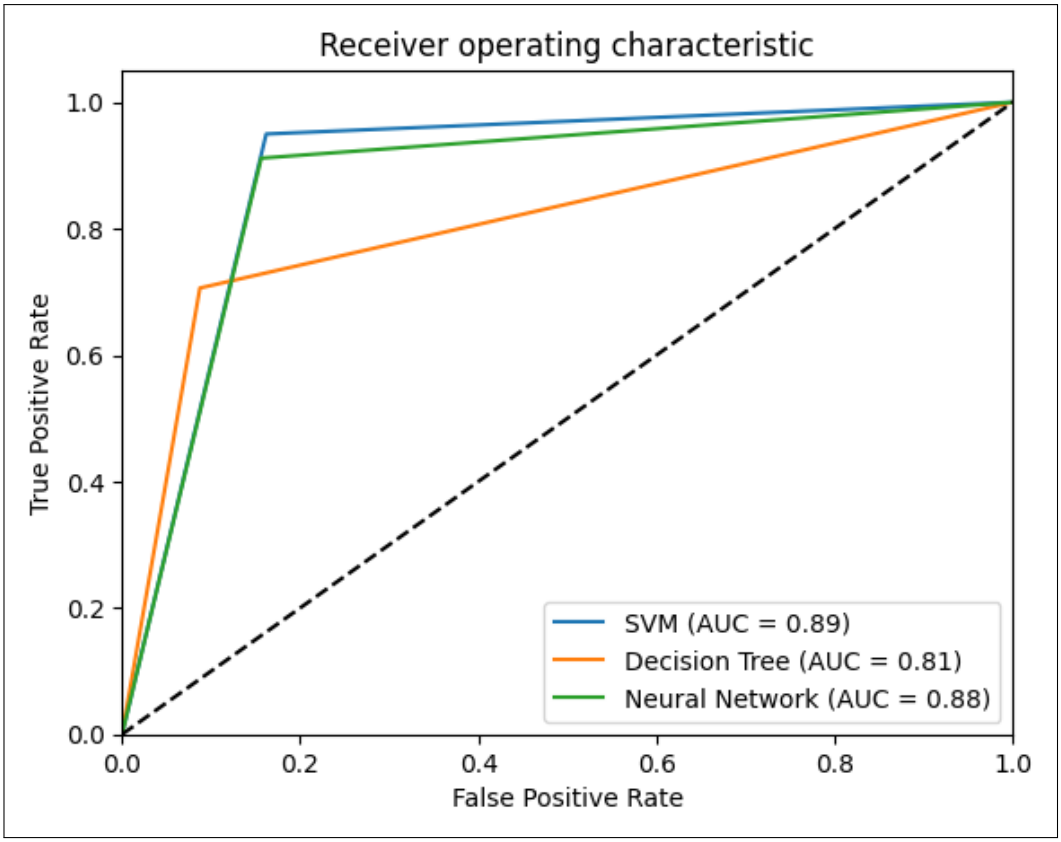

Figura 4.4: Grafico ROC