UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Corso di

Machine Learning & Data Mining

Relazione di progetto:

AI4I Predictive Maintenance: Machine Failure Classification

Esaminandi:

Edoardo Coppola - 719599

Andrea Fiori - 719219

Anno Accademico 2020/2021

Introduzione

La presente relazione ha lo scopo di documentare il lavoro svolto durante il progetto previsto dal corso di *Machine Learning & Data Mining*. Sono illustrate le osservazioni, le analisi e le procedure seguite a partire dalle fasi iniziali e preliminari di *data analysis* fino alla valutazione delle prestazioni dei modelli individuati ed addestrati.

Il *dataset* utilizzato [1] contiene record caratterizzati da misurazioni di grandezze fisiche in ambito industriale e delle etichette che specificano per un guasto o per il corretto funzionamento del macchinario in esame. Il compito previsto dal progetto consiste nell'individuazione di un modello capace di classificare nuove istanze, la cui forma è la medesima dei record sopracitati, come "ad alto rischio di guasto" o come "normale funzionamento". Tale compito è quindi un compito di classificazione binaria. Maggiori dettagli sui record, gli attributi e le etichette relative saranno illustrati in dettaglio nelle sezioni successive.

1. Il dataset: struttura e proprietà

Il *dataset* annovera diecimila istanze, ciascuna caratterizzata da otto attributi riassunti in tabella 1.

Tabella 1 – *Attributi delle istanze all'interno del dataset*

Nome attributo	Descrizione
Product ID	Identificativo di un prodotto della lavorazione. È caratterizzato da una lettera tra L, M, H che ne indica la qualità (bassa, media o alta) seguita da un numero di serie
Type	Il tipo del prodotto realizzato. Coincide con la lettera presente nel *Product ID*
Air Temperature [K]	Temperatura dell'aria, in Kelvin, realizzata tramite generazione di numeri casuali in seguito trasformati in modo da avere una distribuzione gaussiana di media 300 K e deviazione standard di 2 K
Process Temperature [K]	Temperatura di processo generata casualmente, normalizzata con una

	deviazione standard di 1 K e sommata alla temperatura dell'aria più 10
Rotational Speed [rpm]	Velocità di rotazione calcolata a partire da una potenza di 2860 W e sovrapposta ad un rumore gaussiano
Torque [Nm]	Momento torcente i cui valori sono distribuiti in modo normale attorno ad una media di 40 Nm e con una deviazione standard di 10 Nm. Tutti i valori sono non negativi
Tool wear [min]	Usura del macchinario espressa in minuti
Machine Failure [binary]	Etichetta che mostra se si è verificato un guasto o meno. '1' se si è verificato, '0' altrimenti

Il *dataset* è estremamente sbilanciato sicché solamente 339 istanze sono etichettate con '1' (guasti). Questo ha reso molto complicato l'addestramento dei modelli e il raggiungimento di prestazioni dignitose e ha quindi richiesto che venissero prese in considerazioni sia tecniche a livello algoritmico che a livello dei dati stessi. Tali tecniche verranno illustrate nelle sezioni successive.

Per quanto riguarda le proprietà intrinseche dei dati in nostro possesso, possiamo innanzitutto osservare la presenza di alcune forti correlazioni tra coppie di attributi. Ad esempio si nota una correlazione di 0.88 tra la temperatura dell'aria e quella di processo, mentre esiste una correlazione di -0.88 tra la velocità di rotazione e il momento torcente. Tuttavia, questi legami non destano troppo scalpore dato che i valori delle due temperature, seppure ottenuti in modo casuale, sono legati matematicamente mentre dalla fisica si sa che il momento torcente è inversamente proporzionale alla velocità rotazionale (in perfetto accordo col significato proprio di correlazione negativa). Possiamo osservare la figura 1 e la figura 2 per avere una visione d'insieme circa le correlazioni tra tutte le coppie di *feature.* Le matrici illustrate nelle figure successive chiaramente sono simmetriche. Dalla figura 1 è poi possibile vedere anche una distribuzione dei valori della temperatura dell'aria, di processo e del momento torcente *simil-gaussiana* proprio come ci si aspettava.

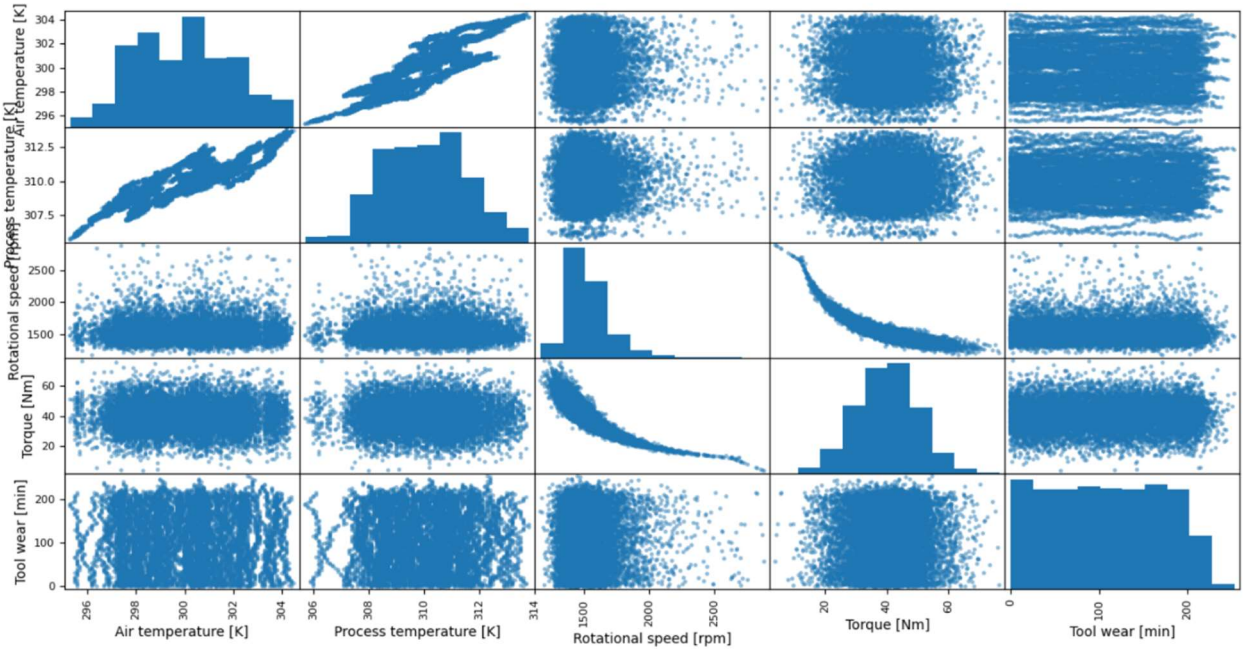

Figura 1 – Matrice delle correlazioni in formato scatter fra tutte le possibili coppie di attributi

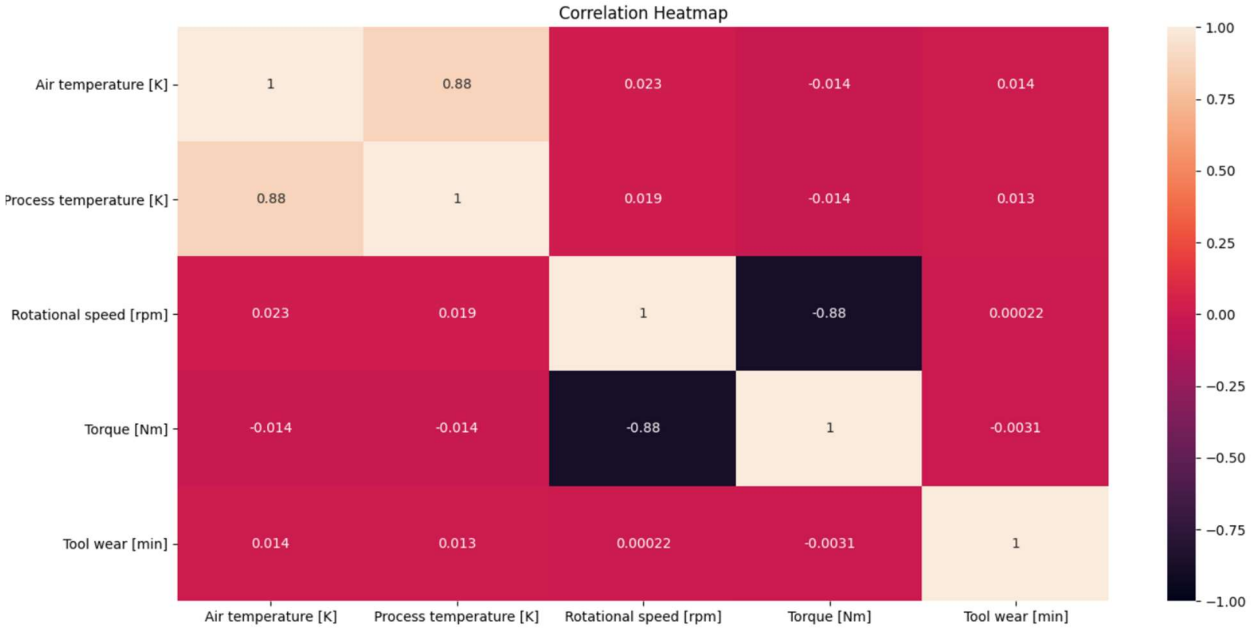

Figura 2 – Heatmap delle correlazioni fra tutte le possibili coppie di attributi

2. Fase preliminare di Data Analysis

In questa fase preliminare ci siamo preoccupati della qualità dei dati, della presenza di record duplicati, di eventuali valori non definiti o mancanti, dell'eventuale presenza di *outliers* e di una prima selezione degli attributi rilevanti ai fini del compito da svolgere.

Per prima cosa è stata eliminato l'attributo *Product ID* perché si compone di una parte irrilevante (*serial number*) e di una ridondante (*type*). Difatti, la qualità di un dato prodotto, espressa tramite una lettera fra L, M o H, è già specificata dall'attributo *Type* dello stesso

record. Il numero di serie del prodotto realizzato, invece, non è rilevante nel caratterizzare un eventuale guasto del macchinario perché non si conosce il prodotto fisico dietro tale numero.

Il *dataset* originale annoverava anche altri cinque attributi (non riportati in tabella 1) caratterizzanti la tipologia di guasto. Per esempio, l'attributo *HDF* (*Heat Dissipation Failure*) specificava per un guasto dovuto al raggiungimento di elevate temperature ed era stato aggiunto, così come per gli altri quattro, per compiti di *Causal Discovery* che esulano dallo scopo di questa relazione. Per questi motivi, tali attributi, e le relative colonne, sono state eliminate.

Stando a quanto riporta l'articolo [2] associato al *dataset*, reperire sufficienti quantità di dati per compiti di manutenzione predittiva è un'impresa ardua. Infatti, spesso simili informazioni non sono note oppure non sono rese pubbliche. Per questo motivo i record utilizzati dall'autore originale del progetto, e dagli scriventi, hanno una natura spesso artificiale o semi-artificiale. Essi, infatti, provengono dalla condensazione di informazioni ed esperienze accumulate in anni di ricerche su questa tematica. Posta quindi la natura dei dati, la ricerca da noi effettuata per trovare record duplicati non ha prodotto alcun risultato. In modo del tutto analogo, e probabilmente per le stesse ragioni, anche la ricerca di valori non definiti o mancanti non ha prodotto alcun risultato. Questi ultimi due esiti sono quindi rassicuranti circa la completezza dal *dataset.*

Per verificare la presenza di valori anomali, o *outliers*, sarebbe necessario disegnare un grafico e riportarvi all'interno le istanze del *dataset*. In questo modo sarebbe possibile individuare quali di esse occupano posizioni "anomale" e quindi scartarle. Tuttavia, la dimensionalità dei record non consente una facile rappresentazione pittorica. Di conseguenza, si è optato per una ricerca grafica di valori anomali nei singoli attributi e, come possono confermare i grafici in figura 3, non ne sono stati trovati. L'attributo di categoria *Type* è stato escluso da questa ricerca perché il grafico avrebbe semplicemente riportato una successione di punti etichettati con L, M, o H. Dai grafici sulle temperature dell'aria e di processo non si notano discontinuità marcate né punti di massimo o minimo eccessivamente rilevanti. Questo può essere dovuto alla trasformazione dei dati casuali originali tramite la distribuzione gaussiana accennata in precedenza. Per quanto riguarda la velocità di rotazione, si nota subito che la presenza di punti di massimo o di minimo è in perfetta contrapposizione con la presenza di punti di minimo e di massimo, rispettivamente, nei valori del momento torcente (come ci si aspetta dato che sono grandezza inversamente

proporzionali e hanno una forte correlazione negativa). La presenza di valori al di fuori della fascia [1500, 2100] rpm e [20, 60] Nm non è da considerarsi negativa, anzi, è proprio ciò che garantisce rappresentatività all'intero *dataset*. Infatti, spesso elevate velocità di rotazione, combinate anche un'elevata usura del macchinario, sono la causa della rottura dello stesso. Quindi, eliminare simili valori sarebbe un grave errore. L'andamento assunto dai valori di usura (*Tool Wear)* è invece molto regolare.

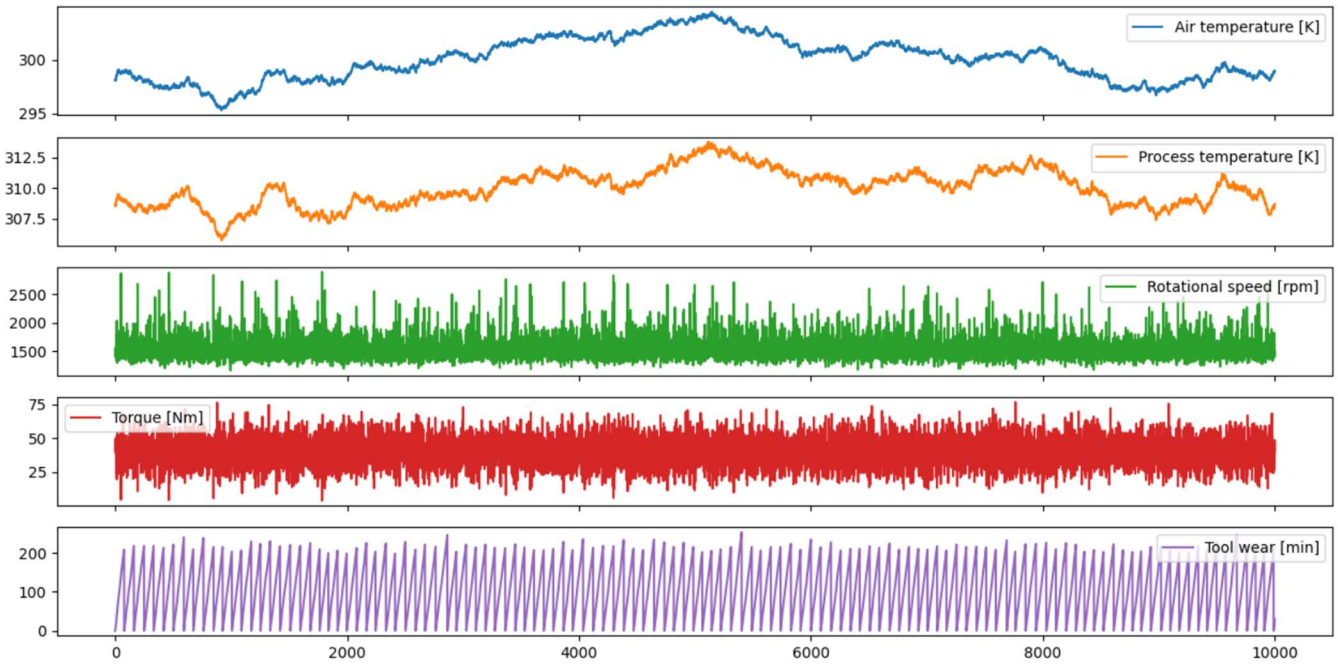

Figura 3 – Andamento dei valori assunti dagli attributi numerici di tutti i record presenti

3. Pre-processing dei dati

Per quanto concerne le misure adottate nel *pre-processing*, abbiamo optato per una codifica *OneHot* dell'attributo *Type* per via dei modelli che avremmo addestrato in seguito. Infatti, sebbene alberi di decisione o *random forest* ben si prestano a lavorare anche con attributi non numerici, non si può dire lo stesso per modelli come *support vector machines*. È stata scelta questa codifica e non una *integer encode* per migliorare l'efficienza e le prestazioni dell'addestramento.

Un altro accorgimento è stato normalizzare le scale degli attributi per motivi di prestazioni sia nella velocità di allenamento che negli *score* dei modelli sui dati di *testing*. A tal fine è stata utilizzata la libreria *StandardScaler* di *sklearn*. L'idea di fondo è quella di realizzare una distribuzione, per ogni *feature* e in modo indipendente, che abbia media nulla e deviazione standard unitaria prima dell'addestramento di qualunque modello. Questo, oltre

a mappare i valori di ogni attributo su una scala comune, rende più veloce l'addestramento di numerose classi di stimatori e può incrementarne le prestazioni.

4. Addestramento dei modelli

4.1 Albero di decisione

Il primo modello che abbiamo deciso di allenare è uno tra i più semplici e noti in letteratura per i compiti di classificazione: un albero di decisione. Abbiamo messo in atto una strategia di *tuning* degli iper-parametri *max_depth* e *min_samples_leaf* tramite l'oggetto *GridSearchCV*[1] combinato con una k-fold cross-validazione stratificata. Una volta terminata la ricerca, abbiamo addestrato degli alberi di decisione inserendo gli iper-parametri appena individuati, utilizzando l'entropia come misura di impurità e passando per una cross-validazione ulteriore che estraesse il miglior modello possibile fra cinque alberi allenati durante il processo. Il migliore fra tutti è stato poi testato sulle istanze del *testing set* dando discrete prestazioni. Abbiamo calcolato anche la matrice di confusione (sui dati di *testing*) del classificatore notando come le prestazioni, specialmente i falsi negativi, non fossero accettabili. Per tali motivi abbiamo deciso di assegnare un peso maggiore alle istanze positive, quelle della classe minoritaria, seguendo un'euristica per la quale il peso assegnato alle istanze di una classe viene calcolato secondo la seguente formula:

$$PesoIstanzeDiClasse_i = \frac{totaleIstanze}{numeroClassi * istanzeDiClasse_i}$$

È possibile verificare con facilità che le istanze della classe minoritaria assumono un peso maggiore di quelle della classe maggioritaria. Grazie a questo accorgimento, senza nulla togliere alla precedente ricerca degli iper-parametri, le prestazioni sono migliorate nettamente fino ad ottenere la matrice di confusione in figura 4.

[1] Anche nota come *Exhaustive Grid Search*, considera tutte le combinazioni possibili fra tutti gli iper-parametri di cui effettuare il *tuning* selezionando i cosiddetti *best_params_* in base al miglior punteggio ottenuto in fase di cross-validazione (anche stratificata come nel nostro caso)

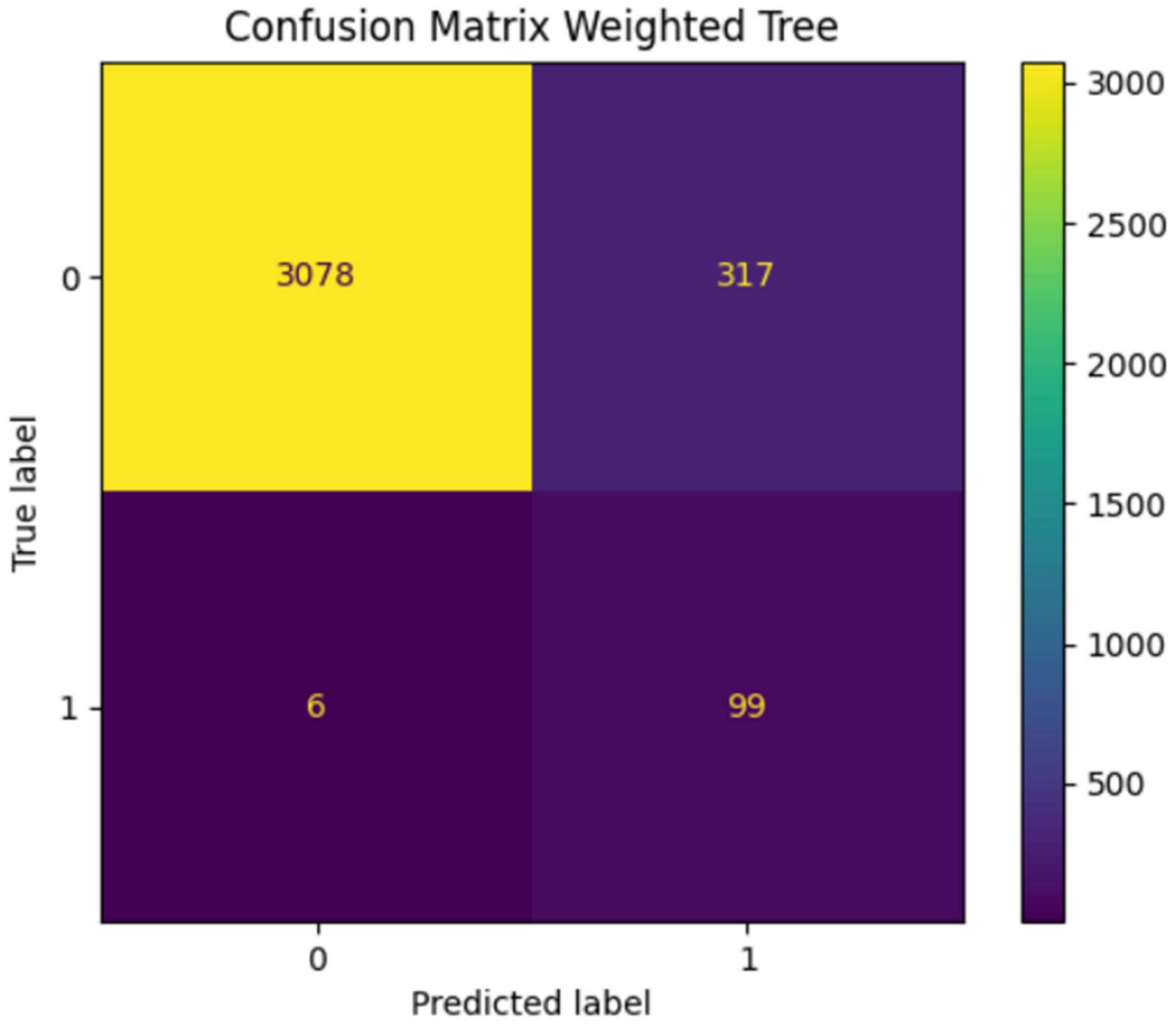

Figura 4 – Matrice di confusione dell'albero di decisione con istanze pesate

Nel tentativo di migliorare ulteriormente le prestazioni abbiamo aggiunto alle tecniche finora illustrate una strategia di *sampling* del *dataset*. In particolare, abbiamo considerato un *oversampling* che aumentasse il numero di istanze della classe minoritaria e un *undersampling* che bilanciasse il numero di istanze delle due classi. Il risultato finale è stato un *dataset* di dimensioni leggermente ridotte ma è rappresentativamente più ricco. La creazione di istanze artificiali è stata realizzata mediante un oggetto *SMOTE*[2]. Il conseguente addestramento ha seguito le stesse fasi di prima:

[2] Aggiunta di istanze sintetiche o artificiali al *dataset* originale così da aumentare il numero degli esempi di classe minoritaria di una quantità a piacere. La sintetizzazione avviene con criterio e buon senso: viene eseguito internamente ai metodi di *SMOTE* una versione di *KNN* che prevede che, qualora siano presenti almeno k istanze della stessa classe entro un raggio ε, ne venga aggiunta una i cui valori delle *feature* siano "simili" a quelli dei vicini

cross-validazione con allenamento di cinque alberi di decisione ed estrazione del migliore con conseguente *testing* finale dello stesso. I risultati ottenuti sono riportati in figura 5.

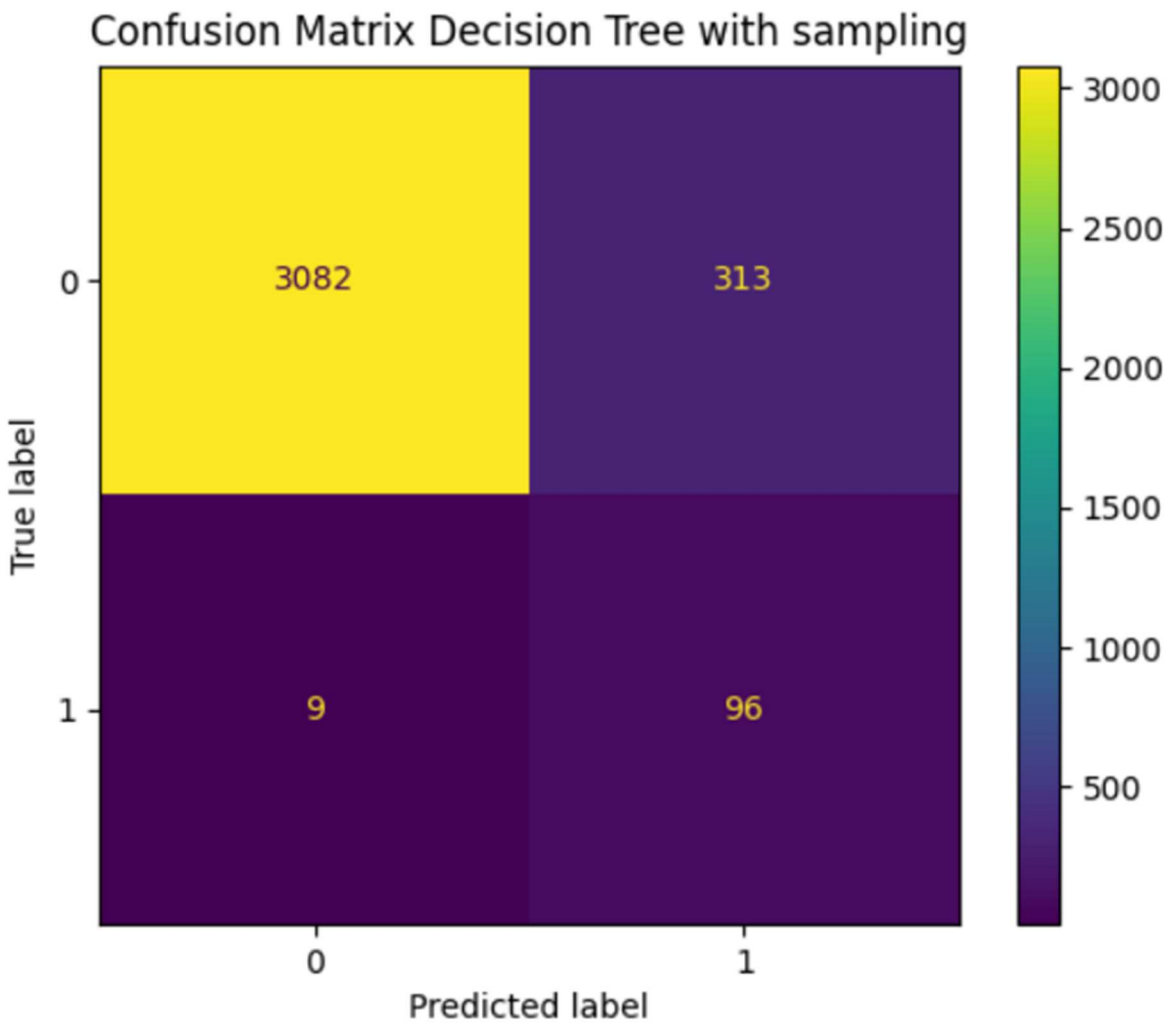

Figura 5 – Matrice di confusione dell'albero di decisione con istanze pesate e sampling sul dataset

Abbiamo quindi osservato che, tra i due modelli finora allenati, il primo registra prestazioni migliori in termini di *FNR*.

4.2 Support Vector Machines

Successivamente siamo passati all'addestramento di una seconda categoria di classificatori: le *support vector machines*. Anche qui abbiamo seguito lo stesso approccio adottato per l'addestramento degli alberi di decisione: una prima serie di modelli allenati con il solo accorgimento dei pesi sulle istanze e una seconda serie di classificatori allenati combinando pesi e *sampling* dei dati.

Il primo *training*, una *SVM* con *kernel RBF*, ha previsto una fase di ricerca degli iper-parametri attraverso *GridSearchCv* con cross-validazione stratificata, illustrata in precedenza. Gli iper-

parametri di cui si è effettuato il *tuning* sono stati *C* e *gamma*. A causa del forte sbilanciamento del *dataset*, la ricerca non ha prodotto risultati soddisfacenti in quanto, fra i tanti modelli vagliati, quello con prestazioni migliori era sempre quello con il massimo numero di falsi negativi. Quest'ultimo era contraddistinto dai valori minimi di *C* e *gamma* considerati dalla ricerca. A tale scopo sono stati impostati manualmente dei valori capaci di migliorare anche minimamente le prestazioni e sono stati allenati, tramite cross-validazione, cinque modelli. Tra questi è stato estratto il migliore le cui prestazioni sui dati di *testing* sono riportate in figura 6.

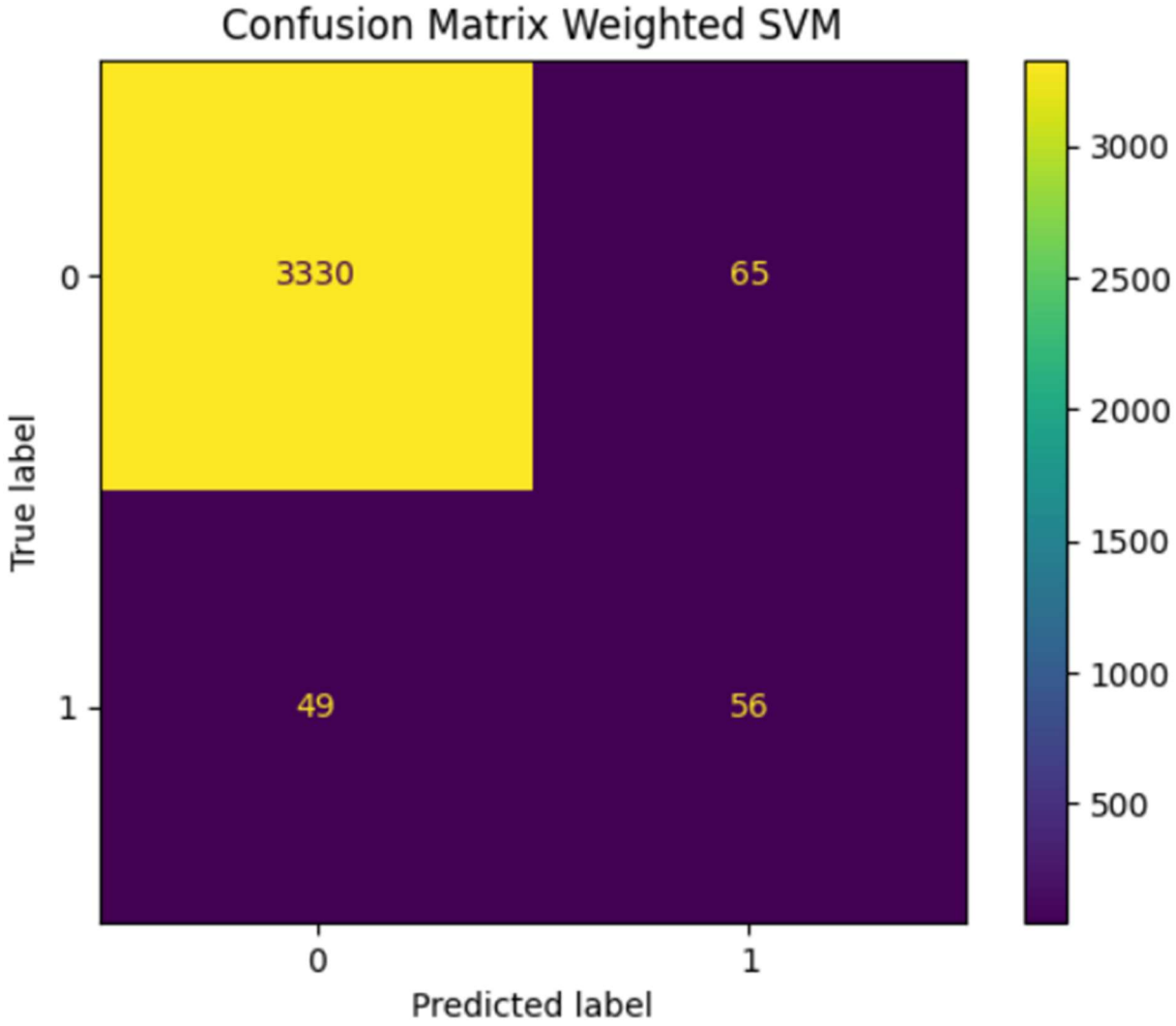

Figura 6 – Matrice di confusione di una SVM con pesi sulle istanze

Nel tentativo di migliorare ulteriormente il *FNR*, abbiamo deciso di allenare una seconda *SVM* combinando agli accorgimenti introdotti in prima battuta anche un *sampling* dei dati ad opera di *SMOTE* e *RandomUnderSampler*. È stata rieseguita la ricerca degli iper-parametri ottimali e, una volta trovati, si è ripetuta la cross-validazione per cercare il migliore tra cinque modelli. Le prestazioni di quest'ultimo sono riportate in figura 7 dove possiamo notare un miglioramento rispetto alla *SVM* precedente. Tuttavia, le prestazioni sono ancora ben lontane dall'essere accettabili in quanto trentanove falsi negativi in un contesto industriale descrivono una situazione di potenziale pericolo.

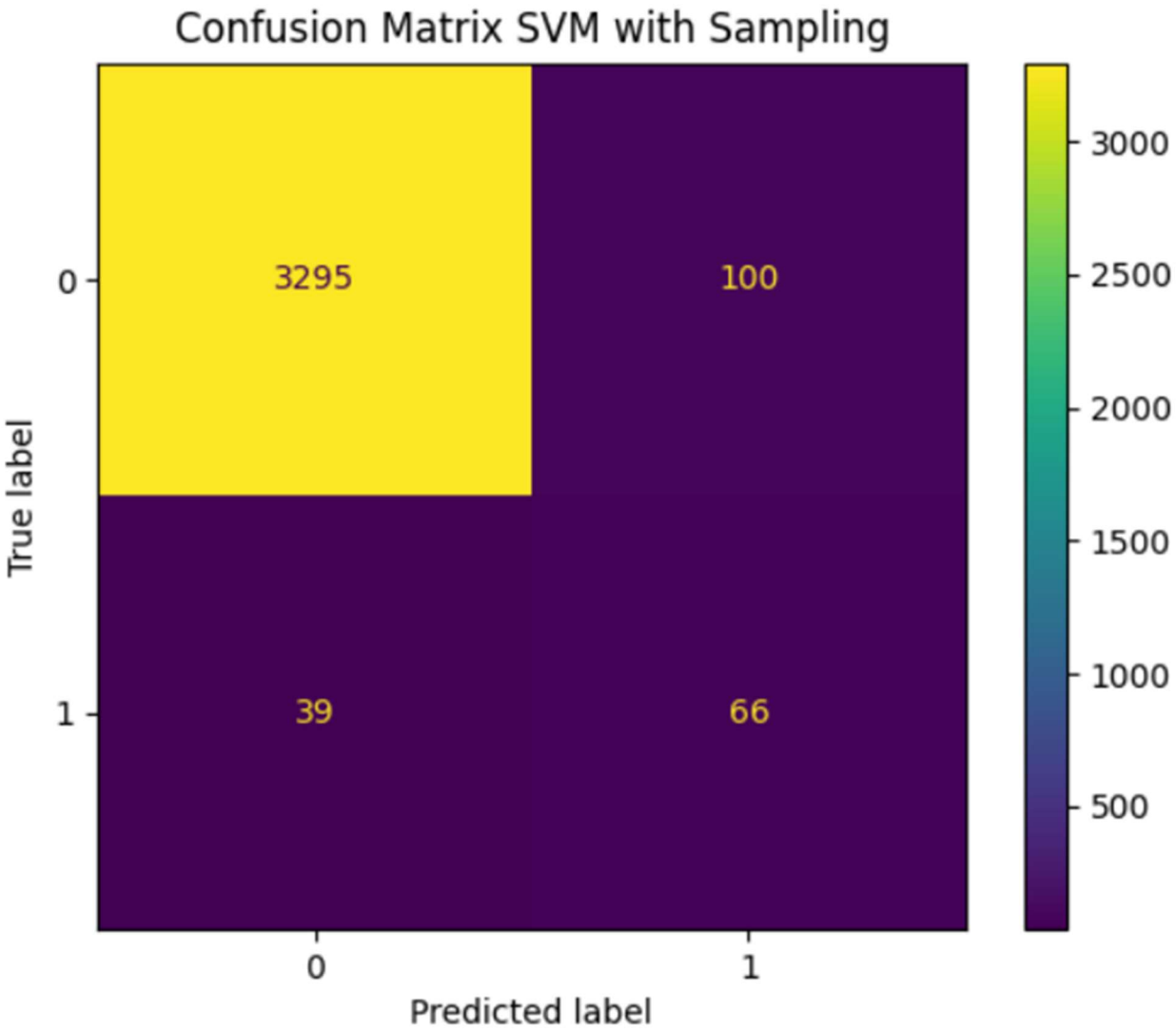

Figura 7 – Matrice di confusione di una SVM con pesi sulle istanze e sampling sui dati

4.3 Classificatore Bayesiano Gaussiano Naive

La classe di modelli addestrati successivamente è quella dei *classificatori bayesiani*. Un presupposto al loro utilizzo consiste nella condizionale indipendenza degli attributi delle istanze che compongono il *dataset*. In altre parole, i valori assunti da un attributo non devono essere influenzati né devono influenzare i valori assunti da altri attributi. Questa precondizione è molto forte e per questo spesso non è soddisfatta nei contesti applicativi. Tuttavia, malgrado questa ipotesi non sia spesso verificata, le prestazioni dei *classificatori bayesiani naive* sono comunque buone e accettabili in numerosi contesti. Per questo motivo abbiamo deciso di allenarne alcuni nonostante l'evidente relazione matematica presente tra varie coppie dei nostri attributi. All'interno di questa categoria di classificatori se ne distinguono altre: quella dei *classificatori bayesiani gaussiani* è tra queste.

L'utilizzo di simili modelli è legato al soddisfacimento di una condizione che prevede che gli attributi abbiamo una distribuzione quanto più gaussiana e questo è proprio il nostro caso. Per quanto

riguarda l'addestramento di simili classificatori, in prima battuta ne abbiamo allenato uno senza considerare alcun tipo di accorgimento al di fuori della solita cross-validazione seguita anche per il *training* dei modelli precedenti. I risultati ottenuti sono stati abbastanza deludenti e sono riportati in figura 8.

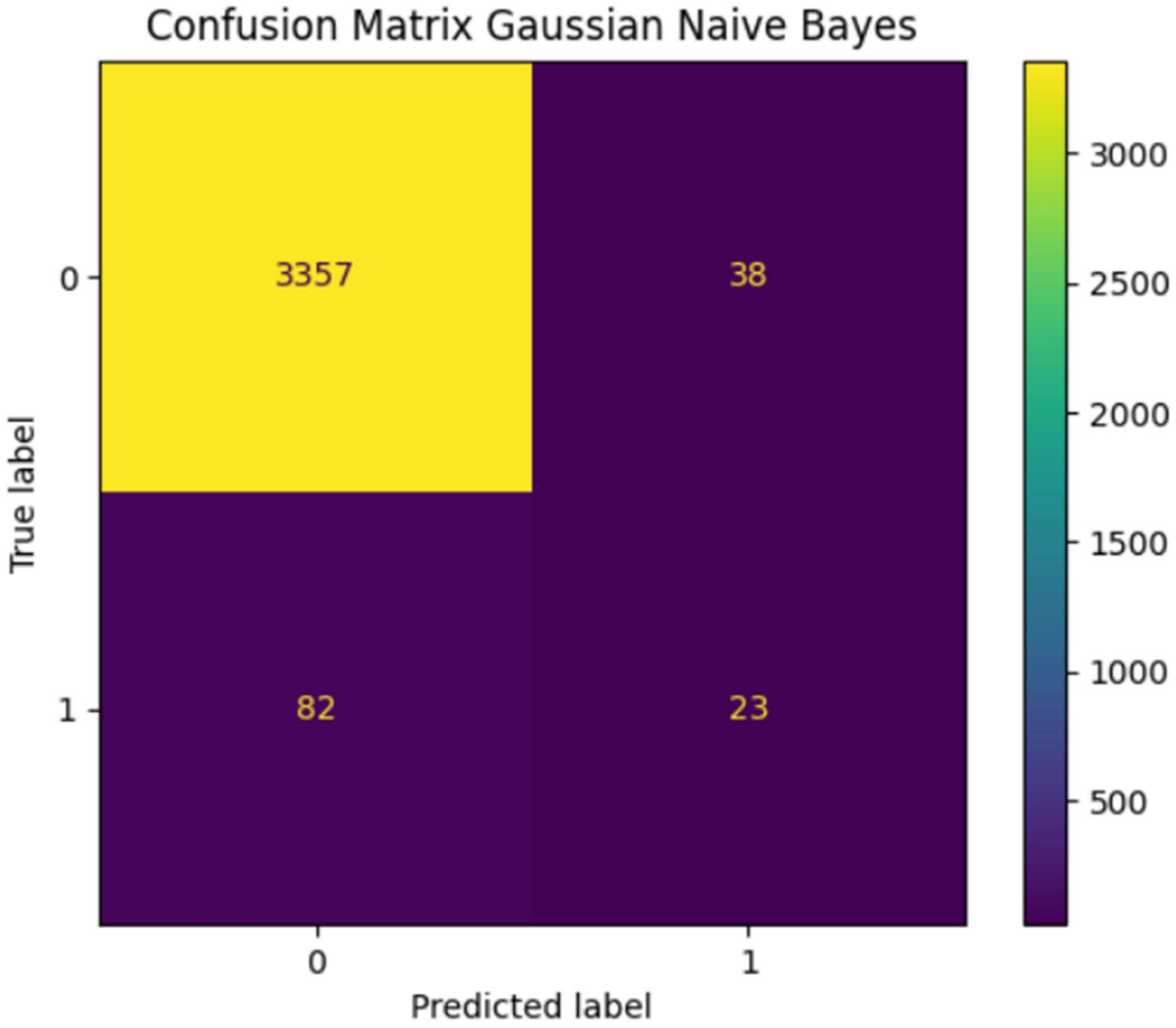

Figura 8 – Matrice di confusione di un classificatore bayesiano naive gaussiano

In un secondo momento abbiamo considerato l'allenamento di un nuovo *classificatore bayesiano* preceduto però da un *sampling* dei dati. Le prestazioni sui dati di *testing* sono migliorate significativamente ma non abbastanza da superare quelle del miglior albero di decisione visto in precedenza. I risultati sono mostrati in figura 9.

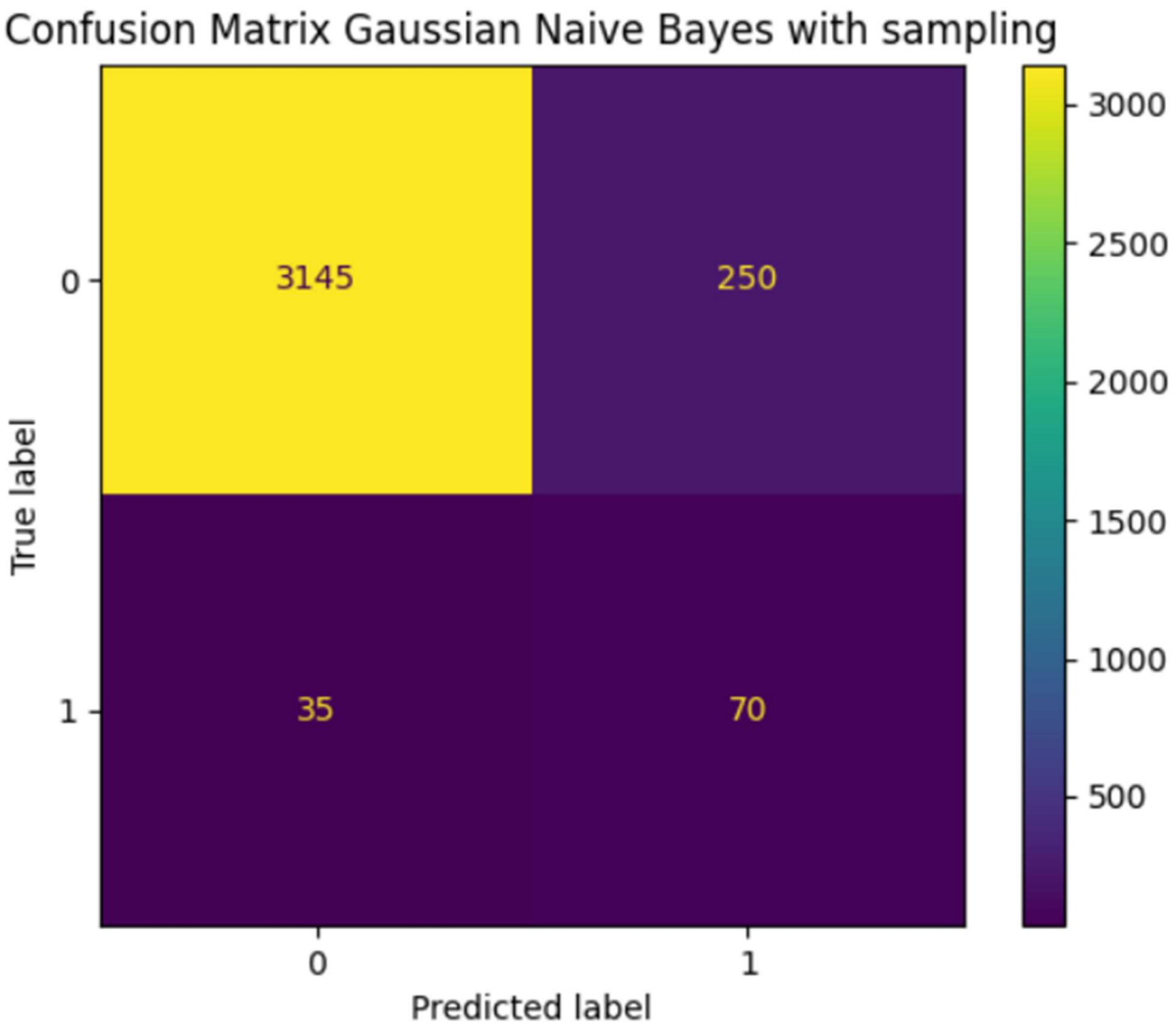

Figura 9 – Matrice di confusione di un classificatore bayesiano gaussiano naive con sampling sui dati

4.4 Rete Neurale

Il nuovo modello che abbiamo addestrato per questo compito è una rete neurale. In questo caso l'addestramento ha subito previsto il *sampling* dei dati al fine di guadagnare rappresentatività e migliorare l'apprendimento dei pesi della rete. È stata anche condotto il *tuning* dei iper-parametri quali *n_iter_no_change*, *max_iter*, *alpha* e *activation*. Il primo[3] rappresenta il numero massimo di epoche consecutive eseguibili senza ottenere un significativo abbattimento dell'errore di validazione; il secondo indica il numero massimo di epoche eseguibili durante il *training*; il terzo è un parametro di regolarizzazione che serve per limitare potenziali situazioni di *overfitting*; infine il quarto è il

[3] Quando viene raggiunto il limite indicato da *n_iter_no_change* si verifica la cosiddetta condizione di *early_stopping* per la quale il *training* viene arrestato. Il motivo del verificarsi di una simile condizione è dovuto probabilmente alla presenza di un minimo locale.

parametro che specifica per la funzione di attivazione. La rete neurale ottenuta alla fine del *training* e di un processo di cross-validazione presenta le prestazioni riportate in figura 10.

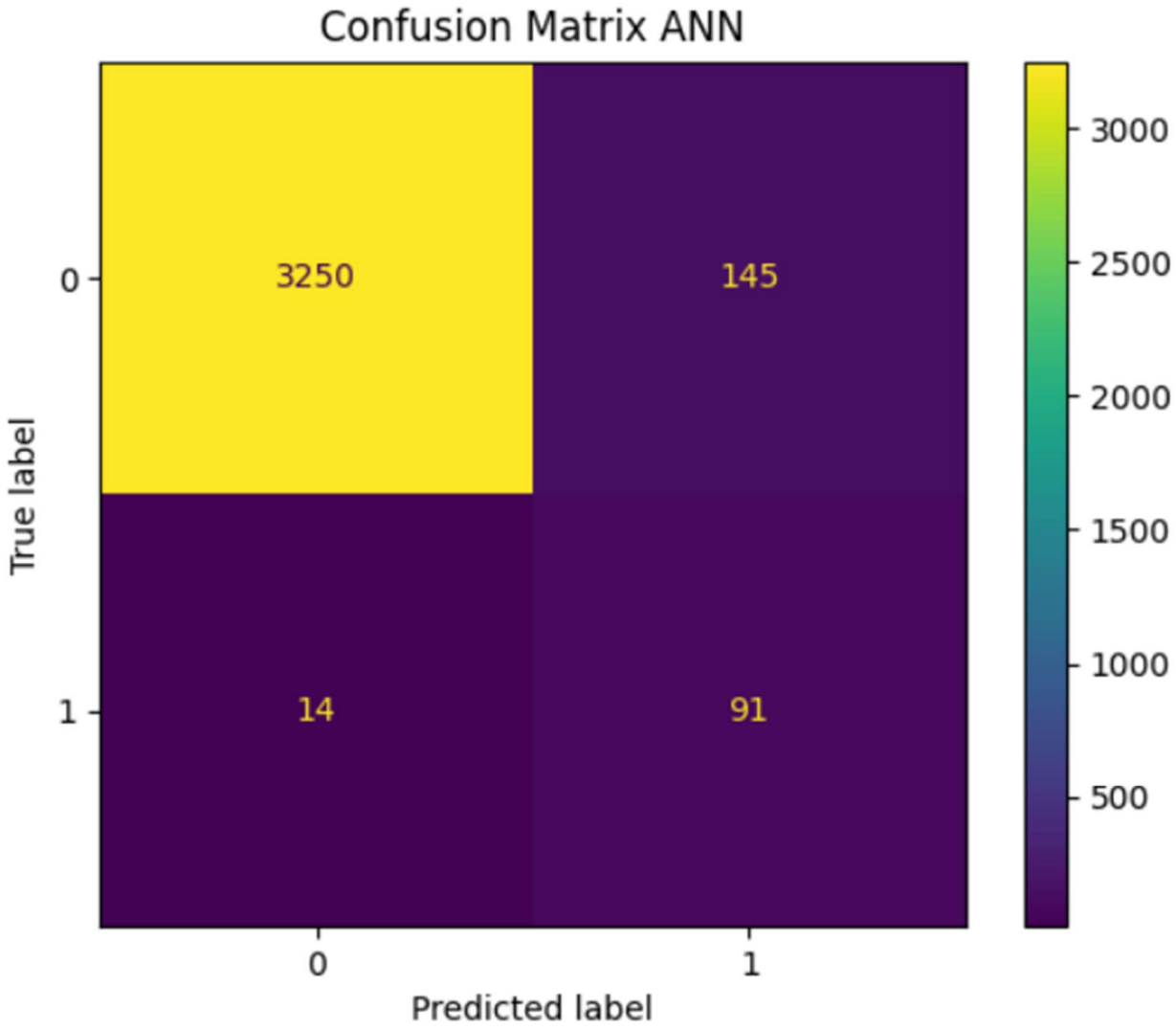

Figura 10 – Matrice di confusione di una rete neurale

Possiamo vedere come le *performance* siano più simili a quelle migliori registrate con gli alberi di decisione sebbene non siano sufficienti a promuovere la rete neurale come miglior modello in assoluto.

4.5 Random Forest

Successivamente siamo passati all'addestramento di un modello *ensemble* come una *random forest*. Qui abbiamo condotto il *tuning* di iper-parametri come *n_estimators* e *max_samples*: il primo indica il numero di stimatori che compongono il modello *ensemble*; il secondo indica la percentuale di *training set* da utilizzare per l'addestramento dei singoli alberi di decisione. Il *tuning* è stato condotto sempre con *GridSearchCV* combinata con una cross-validazione stratificata. Si è scelto,

poi, di dotare le istanze della classa minoritaria di un peso maggiore rispetto a quelle della classe maggioritaria. In questo modo abbiamo fatto sì che il differente peso dei record delle due classi fosse preso in considerazione nell'addestramento dei singoli alberi. Altri importanti iper-parametri che sono stati considerati nell'addestramento degli alberi di decisione, come *max_depth* o *min_samples_leaf*, non sono stati impostati questa volta per evitare che venissero adottate tecniche di *pruning* sugli alberi della foresta. Anche questa volta il *training* è passato per una cross-validazione dalla quale è stato estratto il miglior modello possibile. Le prestazioni che si sono registrate sono illustrate nella matrice di confusione in figura 11.

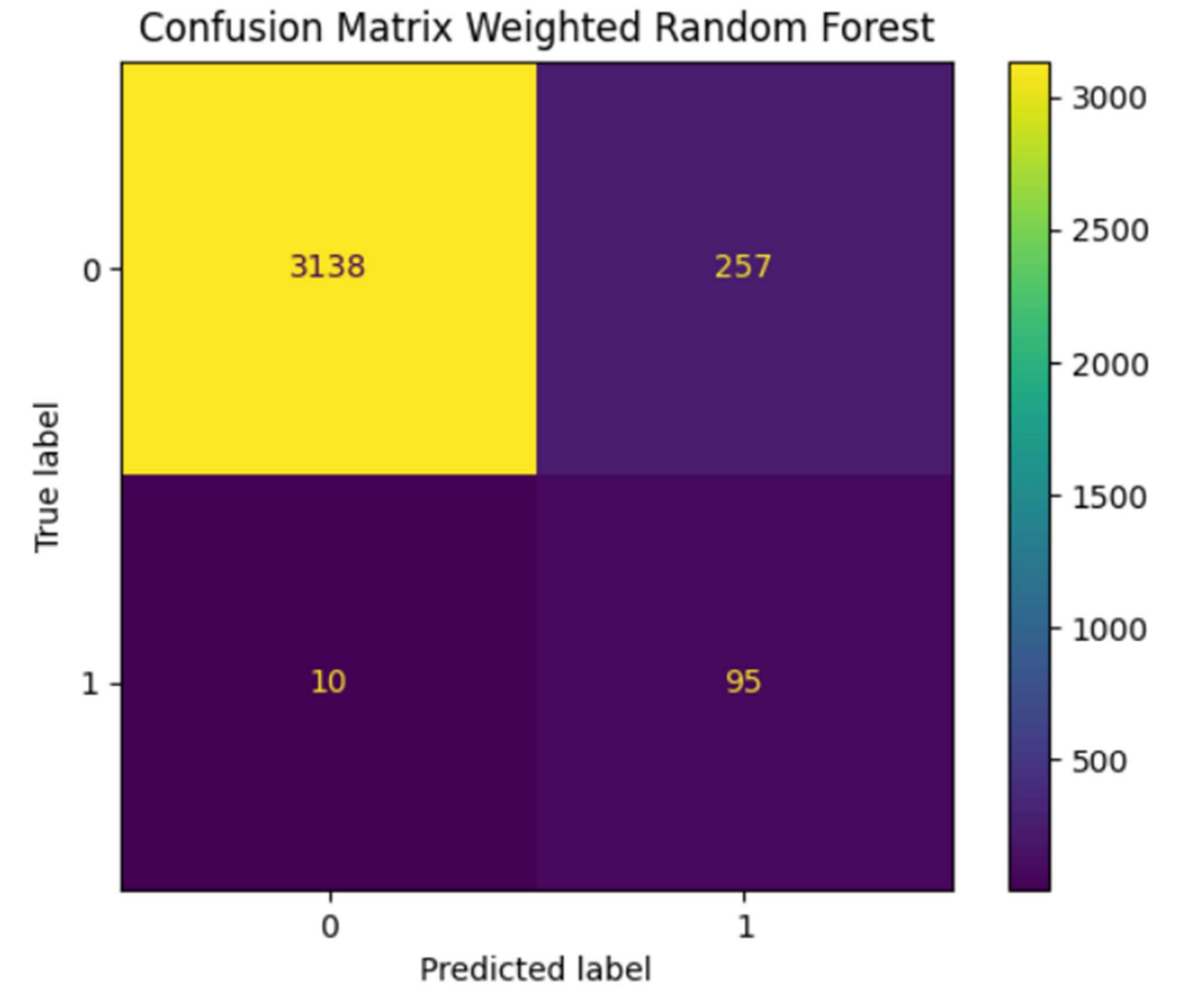

Figura 11 – Matrice di confusione di una random forest con pesi sulle istanze

Possiamo notare che le prestazioni siano migliori di quelle ottenute con la *ANN* ma leggermente peggiori di quelle ottenute con gli alberi di decisione in termini di *FNR.*

Come già visto in precedenza, il passo successivo è stato quello di considerare anche una strategia di *sampling* dei dati di *training* prima dell'addestramento di una seconda *random forest* (anche questa cross-validata). I risultati ottenuti sono stati migliori di quelli registrati con la foresta

precedente e sono riportati in figura 12. Notiamo anche che quest'ultimo modello *ensemble* presenta un numero di falsi negativi uguale a quella del miglior modello trovato finora (albero di decisione con pesi sulle istanze).

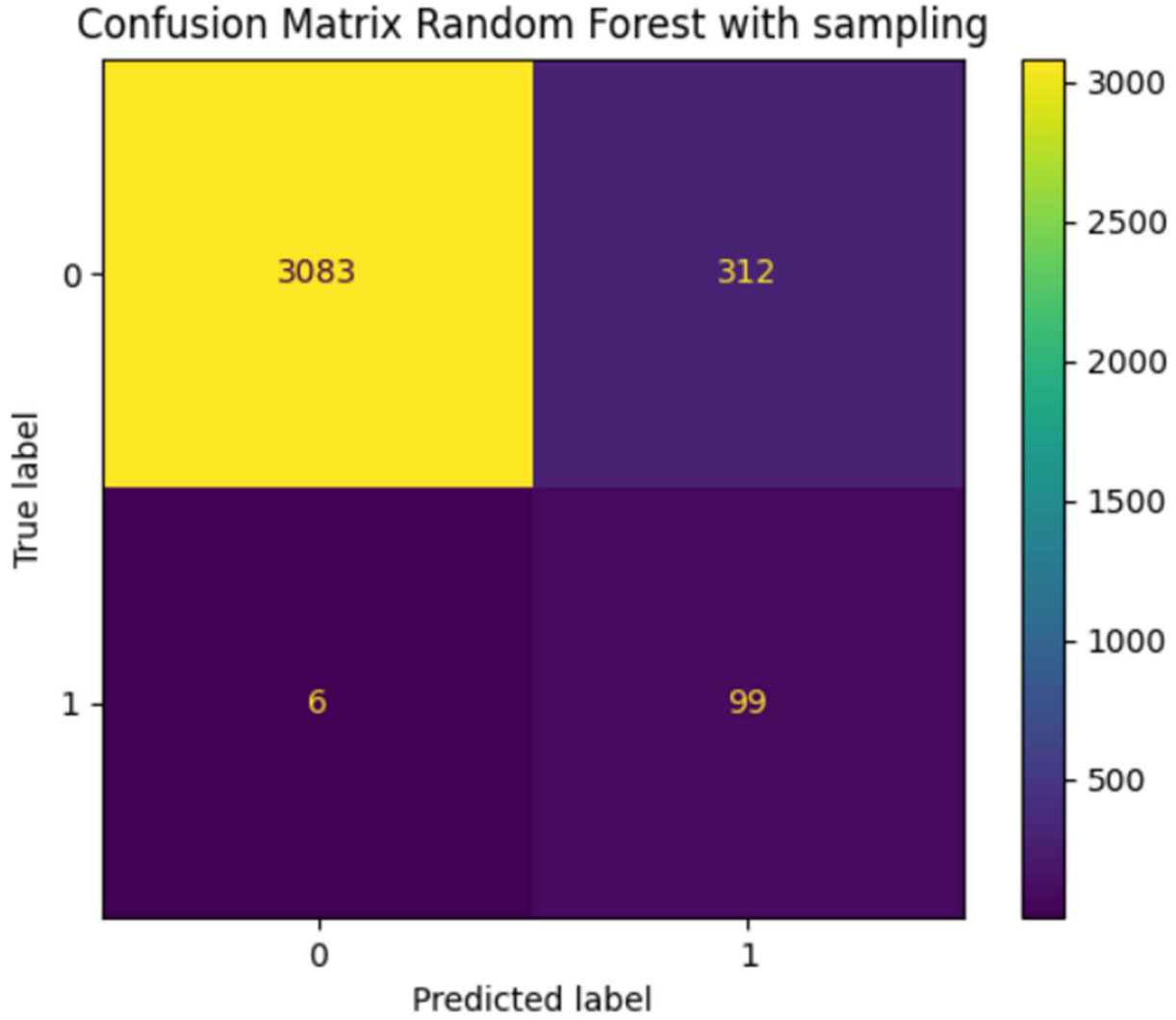

Figura 12 – Matrice di confusione di una random forest con pesi sulle istanze e sampling sui dati

Da quest'ultima foresta abbiamo voluto estrapolare anche la *feature importance* degli attributi utilizzati fino a questo punto. Possiamo notare che gli attributi di maggiore importanza siano la velocità rotazionale, il momento torcente e, infine, l'usura del macchinario. Per questo motivo può essere utile riaddestrare il modello migliore considerando unicamente questi tre attributi e verificare l'eventuale raggiungimento di migliori prestazioni.

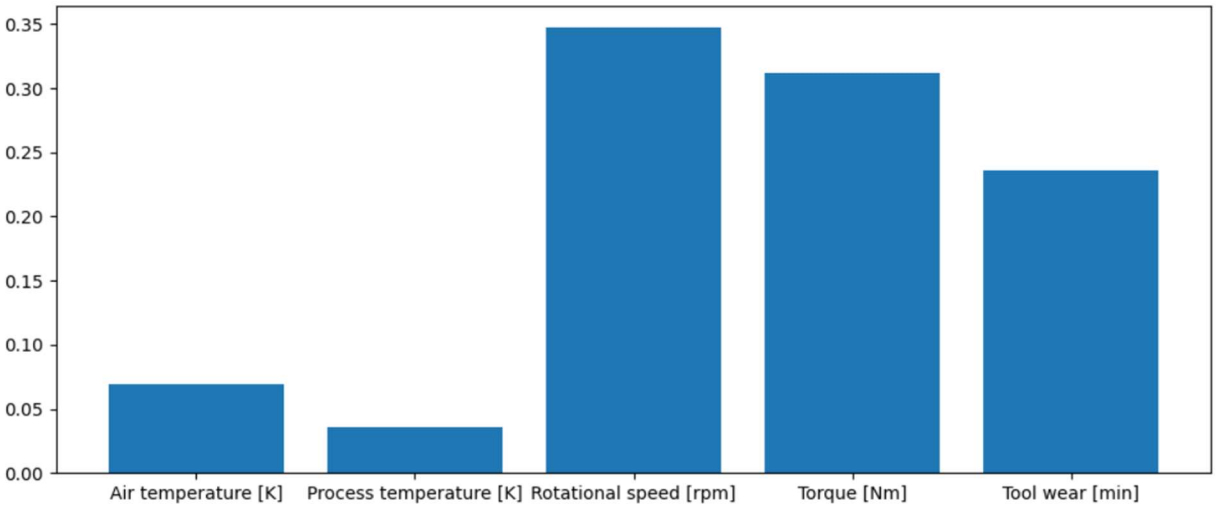

Figura 13 – Importanza delle feature

Riaddestrando il miglior albero di decisione, con gli stessi iper-parametri individuati in precedenza, si ottengono prestazioni tali da rendere quest'ultimo modello il migliore individuato, come si può osservare dalla matrice di confusione in figura 14.

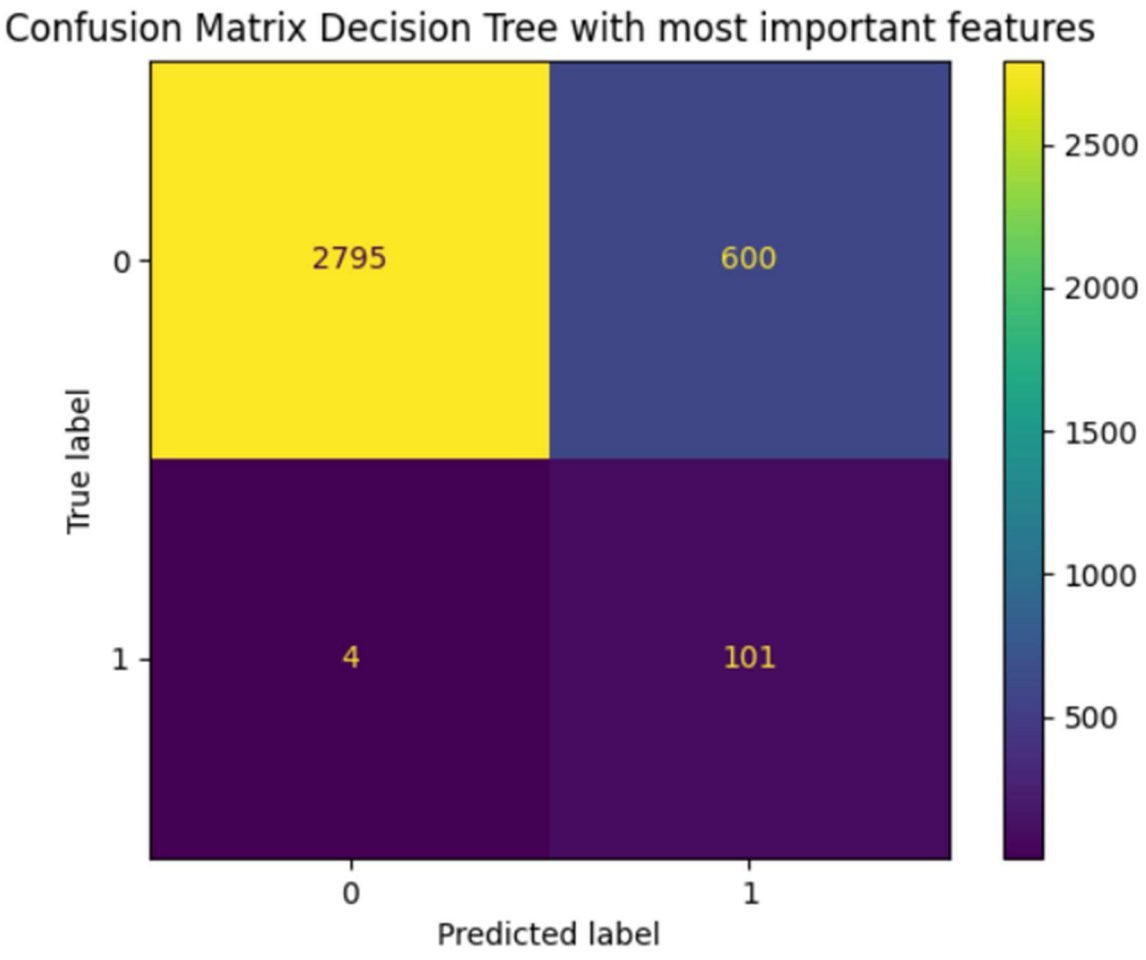

Figura 14 – Matrice di confusione del miglior albero di decisione allenato con solo gli attributi più importanti

4.6 AutoSkiLearn Model

Infine, un ultimo tentativo è stato svolto utilizzando il *tool* automatico fornito da *Ski-Learn*. Il *dataset* dato in input a questo classificatore ha subito un *sampling* e le prestazioni che si sono ottenute sono riportate in figura 15.

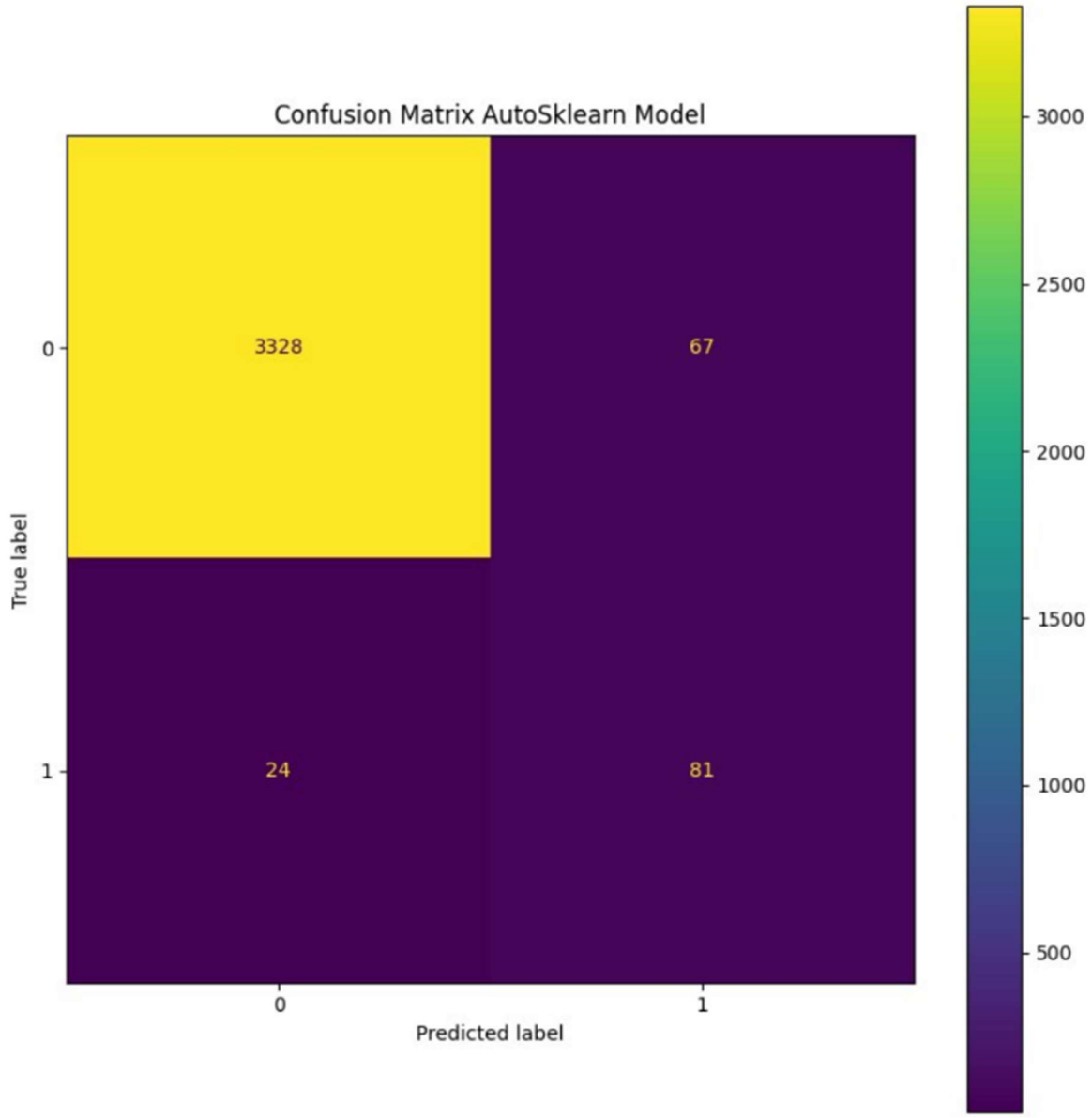

Figura 15 – Matrice di confusione del modello Auto Ski-Learn

Possiamo notare come le prestazioni offerte dal *tool* automatico non siano migliore di quelle ottenute in precedenza.

5. Conclusioni

In conclusione possiamo osservare che lo sbilanciamento del *dataset* è stato il fattore di maggiore difficoltà nell'addestramento dei modelli. Gli accorgimenti a livello dei dati (*over* e *undersampling*) e quello a livello algoritmico (l'aggiunta di pesi alle istanze) hanno giocato un ruolo fondamentale nell'apprendimento di classificatori efficienti. In particolare, gli alberi di decisione e le *random forest* sono quelli che hanno fornito le migliori *performance* in assoluto. Al contrario gli altri modelli non hanno saputo fornire risultati, in termini di *false negative rate*, che fossero quantomeno confrontabili coi primi.

Il modello finale da scegliere è in assoluto l'albero di decisione addestrato unicamente sui tre attributi più importanti: la velocità rotazionale, il momento torcente e l'usura del macchinario. Questo albero di decisione non è solo il classificatore che ha ottenuto il minor *FNR*, ma è anche estremamente comprensibile, snello e dal ridotto numero di nodi. Una sua rappresentazione è fornita dalla figura 16.

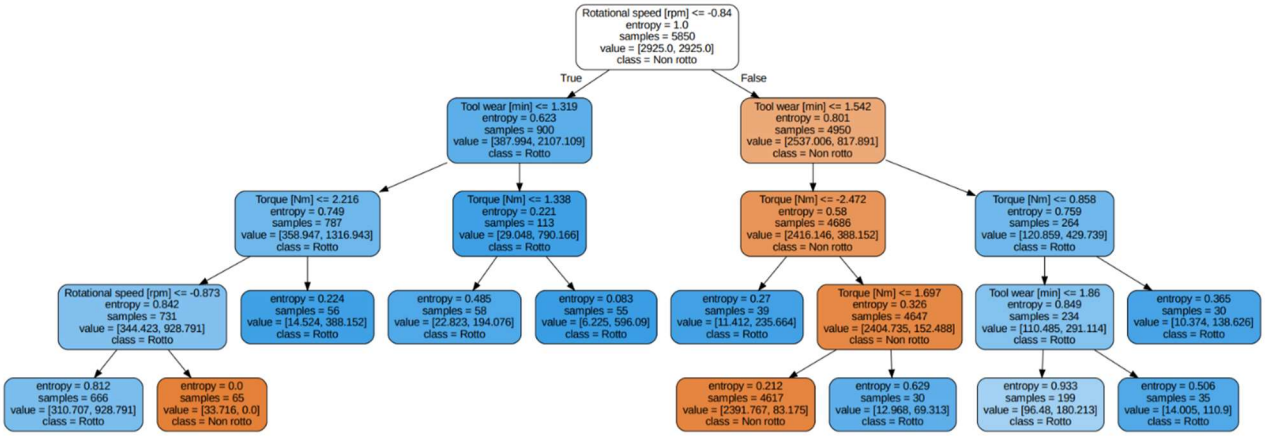

Figura 16 – Albero di decisione allenato sulle tre feature più importanti

Riferimenti

[1]: https://archive.ics.uci.edu/ml/datasets/AI4I+2020+Predictive+Maintenance+Dataset

[2]: Explainable Artificial Intelligence for Predictive Maintenance Applications', Third International Conference on Artificial Intelligence for Industries (AI4I 2020), 2020