Progetto di *Tecniche di Data Mining*

Matteo Mastroianni, Matteo Cantafio
Dipartimento di Ingegneria Meccanica, Energetica e Gestionale
Università della Calabria

CONTENTS

LIST OF FIGURES

Progetto di *Tecniche di Data Mining*

***Abstract*—Il rilevamento delle frodi nelle transazioni con carte di credito è un problema complesso a causa della natura sbilanciata dei dati, caratterizzati da una percentuale minima di transazioni fraudolente rispetto al totale. Questo studio propone un framework per l'analisi e la classificazione delle transazioni, utilizzando tecniche di preprocessing e selezione delle feature. Sono stati esplorati due approcci distinti: uno che prevede l'undersampling preliminare e un altro che integra l'undersampling nella procedura di cross-validation per ridurre il rischio di overfitting. I modelli di machine learning, tra cui Support Vector Machine, Neural Network e Alberi Decisionali, sono stati sottoposti nel primo approccio ad una tecnica di Ottimizzazione Bayesiana per migliorare l'Accuratezza e hanno riportato risultati soddisfacenti anche per quanto riguarda le ulteriori metriche di prestazione come l'AUC-ROC, l'F1-score e l'MCC. Questo lavoro rappresenta un primo passo verso un approccio sistematico per affrontare la rilevazione di attività illecite, con potenziali applicazioni pratiche in scenari reali.**

I. Introduzione

Questo progetto nasce dall'esigenza di affrontare una delle sfide più pressanti nel campo della sicurezza finanziaria: la rilevazione delle frodi nelle transazioni con carte di credito. L'obiettivo principale è sviluppare e valutare modelli di apprendimento automatico capaci di distinguere con precisione tra transazioni legittime e fraudolente. La motivazione alla base di questo lavoro risiede nella crescente sofisticazione delle tecniche di frode e nella necessità di proteggere gli utenti da perdite economiche significative. Attraverso l'analisi di un dataset altamente sbilanciato, il progetto si concentra sull'implementazione di metodologie per il pre-processamento dei dati e la selezione delle feature, nonché a proporre soluzioni per migliorare l'accuratezza dei modelli di classificazione.

II. Data Set

Il primo passo fondamentale consiste nell'acquisire una comprensione di base del nostro dataset, composto da *284.807* istanze, 30 features e una classe target di tipo binario. Va ricordato che, ad eccezione delle colonne *Amount* e *Time*, non disponiamo di informazioni specifiche sulle altre 28 features, i cui nomi sono stati omessi per motivi di privacy. Tuttavia, sappiamo che queste ultime sono già state sottoposte a un processo di Scaling. La feature *Time* rappresenta i secondi trascorsi tra ogni transazione e la prima transazione nel dataset. La featura *Amount*, invece, indica l'importo della transazione. Un aspetto tecnico rilevante è che, come indicato nella descrizione del dataset, le features nominate da *V1* fino a *V28* sono il risultato di una trasformazione PCA (Principal Component Analysis), una tecnica che richiede, come prerequisito, che le caratteristiche siano previamente sottoposte a Scaling. Presumiamo, quindi, che le features denominate "V" siano già state opportunamente scalate dai creatori del dataset. Il nostro dataset presenta uno sbilanciamento molto importante per quanto riguarda la distribuzione delle etichette della Classe Target, infatti la maggior parte delle transazioni risulta essere etichettata come 0:"Non Fraudolenta" (99,83%), mentre solo una minima parte (0,17%) è etichettata come 1:"Fraudolenta".

III. Task di Knowledge Discovery

Il problema in analisi si configura come un problema di classificazione binaria, il cui obiettivo è distinguere con precisione le transazioni fraudolente da quelle non fraudolente. Per risolverlo è necessario sviluppare un modello capace di rilevare attività illecite riducendo al minimo il numero di falsi negativi. Tale obiettivo è cruciale, poiché l'identificazione tempestiva delle frodi riveste un ruolo fondamentale nella protezione degli utenti e nella prevenzione di significative perdite economiche. Per affrontare il problema, lo studio è costituito dalle seguenti fasi:

1) Analisi esplorativa dei dati (EDA): dopo aver caricato il dataset sul software Orange abbiamo condotto un'analisi esplorativa dei dati in modo tale da poter visualizzare le distribuzioni delle feature e comprendere le caratteristiche principali del dataset.
2) Selezione e pre-processamento dei dati: abbiamo standardizzando le feature *Amount* e *Time* mediante la tecnica di scaling z-score.
3) Selezione delle feature: sono state selezionate 21 feature in grado di spiegare il 90% della varianza totale.
4) Costruzione dei modelli: Sono stati sperimentati due approcci. Nel primo approccio, il dataset è stato sottoposto a un processo di undersampling, selezionando tutte le 492 istanze fraudolente presenti nel dataset originale e bilanciandole con 800 istanze non fraudolente, per un totale di 1.292 campioni. Successivamente, sono stati addestrati tre modelli di Machine Learning (Rete Neurale, SVM, Albero Decisionale), valutati mediante 5-Fold Cross Validation. Gli iperparametri sono stati scelti tramite l'Ottimizzazione Bayesiana, con l'obiettivo di massimizzare l'accuratezza. Nel secondo approccio, la cross-validation è stata integrata con l'undersampling per mitigare il rischio di overfitting.

IV. Statistica Descrittiva e Visualizzazione

La prima fase del progetto ha riguardato un'analisi esplorativa dei dati (EDA) utilizzando i seguenti widget: Feature Statistics, Distribution, Box Plot e Scatter Plot. Attraverso il widget Feature Statistics, abbiamo appreso che l'importo medio delle transazioni è piuttosto contenuto, attestandosi intorno a 88 dollari statunitensi, e che non sono presenti valori "Null" all'interno del dataset, rendendo superfluo qualsiasi intervento per la gestione dei dati mancanti. Successivamente, il widget

Class Distribution ha fornito informazioni di particolare rilievo, che riportiamo di seguito.

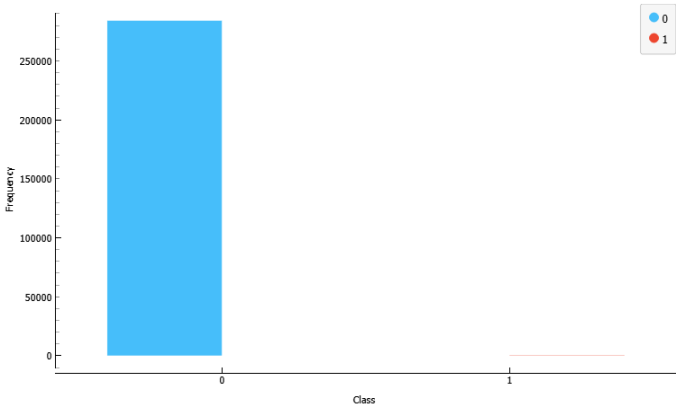

Fig. 1: Class Distribution Plot

Dal grafico emerge chiaramente il marcato squilibrio tra le classi presenti nel dataset, fenomeno che impone un adeguato trattamento per garantire il successo dell'analisi. Successivamente abbiamo analizzato il grafico scatter plot, nel tentativo di individuare pattern e relazioni tra le variabili. Tuttavia, l'elevato numero di feature ha reso difficile una comprensione visiva efficace, spingendoci a focalizzarci su un'analisi della varianza per identificare le feature più rilevanti. Prima di procedere con la Feature Selection, abbiamo effettuato una standardizzazione delle variabili *Amount* e *Time* utilizzando lo z-score, con media pari a 0 e varianza unitaria. Tale intervento si è reso necessario poiché queste ultime presentavano una scala di valori significativamente superiore rispetto alle altre feature, con il rischio di generare problemi di instabilità numerica quali Overflow o Underflow. Successivamente, ci siamo concentrati sulla selezione delle features da includere nel modello. In un primo momento abbiamo implementato in un widget python script un codice che restituisse uno scree-plot delle feature presenti all'interno del dataset, in modo da poter visualizzare in corrispondenza di ogni feature il relativo valore di varianza.

In seguito però, per agevolare la selezione delle feature da mantenere nel dataset, abbiamo creato una nuova datatable contenente 30 righe, una per ciascuna feature ordinate per valori decrescenti di varianza, e tre colonne principali: la prima riporta appunto il valore di varianza in corrispondenza di ciascuna feature, la seconda il valore percentuale della varianza e l'ultima il valore cumulato di varianza espresso in percentuale. Sulla base dei dati contenuti nella colonna della varianza cumulativa, sono state selezionate le feature capaci di spiegare circa il 90% della varianza totale, identificandole nel range compreso tra V1 e V19.

Inoltre, per garantire che le feature selezionate non fossero fortemente correlate tra loro, abbiamo utilizzato il widget Correlations. Questo passaggio è stato cruciale per verificare uno dei risultati fondamentali della PCA, ossia l'indipendenza

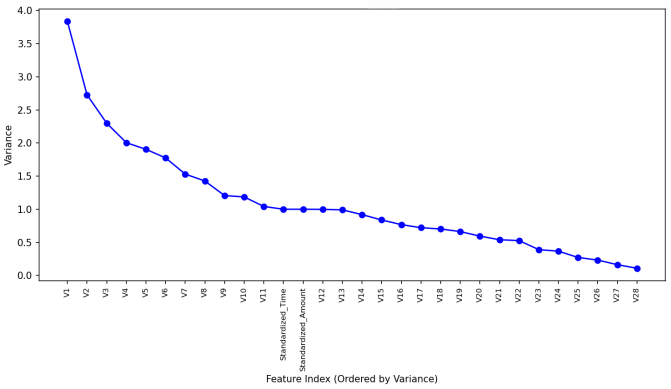

Fig. 2: Scree-plot

Cumulative Variance % for Feature selection

	Feature	Variance	Variance %	Cumulative Variance %
1	V1	3.83649	11.721	11.721
2	V2	2.72682	8.33078	20.0517
3	V3	2.29903	7.02383	27.0756
4	V4	2.00468	6.12457	33.2001
5	V5	1.90508	5.82027	39.0204
6	V6	1.77495	5.42269	44.4431
7	V7	1.5304	4.67557	49.1187
8	V8	1.42648	4.35808	53.4767
9	V9	1.20699	3.68752	57.1643
10	V10	1.18559	3.62214	60.7864
11	V11	1.04186	3.183	63.9694
12	Standardize...	1	3.05514	67.0245
13	Standardize...	1	3.05514	70.0797
14	V12	0.998403	3.05025	73.1299
15	V13	0.990571	3.02632	76.1562
16	V14	0.918906	2.80737	78.9636
17	V15	0.837803	2.5596	81.5232
18	V16	0.767819	2.34579	83.869
19	V17	0.721373	2.20389	86.0729
20	V18	0.702539	2.14635	88.2192
21	V19	0.662662	2.02452	90.2438
22	V20	0.594325	1.81574	92.0595
23	V21	0.539526	1.64832	93.7078
24	V22	0.526643	1.60896	95.3168
25	V23	0.389951	1.19135	96.5081
26	V24	0.366808	1.12065	97.6288
27	V25	0.271731	0.830172	98.4589
28	V26	0.232543	0.710448	99.1694
29	V27	0.162919	0.497739	99.6671
30	V28	0.108955	0.332871	100

Fig. 3: Cumulative Variance %

tra le componenti principali, e quindi confermare che le variabili selezionate fossero effettivamente utili ai fini del modello senza introdurre ridondanze o distorsioni. L'ultima fase del pre-processamento dei dati vede l'impiego della tecnica di Random Undersampling per mitigare l'elevato squilibrio tra le classi. Questa metodologia ha permesso di creare un dataset bilanciato, includendo tutte le transazioni fraudolente del dataset originale e 800 istanze di transazioni non fraudolente selezionate in modo casuale, per un totale di 1.292 istanze. A tal proposito, abbiamo sviluppato due strategie distinte: nella prima, l'undersampling viene eseguito preliminarmente; nella seconda, tale operazione è integrata direttamente con la procedura di cross-validation, garantendo un approccio più dinamico e mirato alla riduzione del rischio di Overfitting.

V. Analisi di Machine Learning

All'interno del nostro spazio di lavoro su Orange abbiamo deciso di implementare 3 diversi modelli di Machine Learning, con lo scopo di compararli a valle della fase di tuning dei rispettivi iperparametri.

Il primo modello è un modello di Support Vector Machine (SVM) con kernel RBF (Radial Basis Function), uno dei modelli più utilizzati per affrontare problemi di classificazione e regressione. Il kernel RBF è una funzione che trasforma i dati in uno spazio ad alta dimensionalità, consentendo al modello di separare i dati non linearmente separabili nel loro spazio originale. E' definito come:

$$K(x, x') = e^{(-\gamma \|x - x'\|^2)} \tag{1}$$

dove $\gamma > 0$ controlla l'influenza di un singolo punto di dati. Un valore elevato di γ rende il modello più sensibile ai singoli data-point, mentre un valore basso aumenta la generalizzazione. Se i dati non sono linearmente separabili, è permesso un certo grado di errore (gestito dal parametro *C*), che bilancia la tolleranza agli errori e la rigidità della separazione.

Per quanto riguarda le reti neurali, queste sono modelli ispirati al funzionamento del cervello umano, composte da neuroni organizzati in strati. Tra gli iperparametri principali, la *struttura della rete*, rappresentata da una lista di numeri interi che identificano il numero di neuroni all'interno di ciascun strato nascosto, influenza la capacità del modello di apprendere rappresentazioni complesse dei dati: reti troppo grandi rischiano di sovradattarsi, mentre reti troppo piccole potrebbero non catturare sufficientemente le relazioni tra le feature. La *funzione di attivazione*, come la ReLU (Rectified Linear Unit), è fondamentale per determinare come il valore di input viene trasformato in output per ciascun neurone, grazie alla sua capacità di evitare il problema del Vanishing Gradient mantenendo i gradienti non nulli per valori positivi. Un altro iperparametro è l'*algoritmo di ottimizzazione*, come ad esempio Adam (Adaptive Moment Estimation), che regola dinamicamente il tasso di apprendimento per ciascun parametro. L'iperparametro di regolarizzazione α, invece, rappresenta un parametro essenziale per controllare la penalizzazione sui pesi, riducendo il rischio di Overfitting e favorendo un modello più generalizzabile. Infine, il *numero massimo di iterazioni* determina il numero di passi compiuti dall'algoritmo di ottimizzazione durante l'addestramento, garantendo che il processo si arresti entro un limite ragionevole.

Gli alberi decisionali, invece, rappresentano modelli intuitivi e facilmente interpretabili che segmentano i dati in base a condizioni su una o più feature. Gli iperparametri principali includono *Min. number of instances in leaves*, che specifica il numero minimo di osservazioni richiesto per creare un nodo foglia dell'albero, con valori maggiori che evitano la creazione di foglie troppo specifiche, migliorando la generalizzazione. L'iperparametro *Do not split subsets smaller than (Min. dimension of subsets)* stabilisce la dimensione minima dei gruppi di dati risultanti da una suddivisione, riducendo il rischio di suddivisioni troppo granulari che potrebbero portare a overfitting. L'iperparametro *Maximal depth* controlla la complessità del modello, limitando il numero di livelli: alberi troppo profondi tendono ad overfitting, mentre alberi troppo bassi potrebbero non riuscire a generalizzare bene. Infine, la soglia di arresto per la classificazione definisce il livello di confidenza necessario per fermarsi in un nodo e fare una previsione, riducendo l'uso eccessivo di ulteriori suddivisioni.

VI. Sperimentazione

Per quanto riguarda il primo approccio, i modelli sono stati addestrati effettuando il random undersampling in fase preliminare, successivamente abbiamo deciso di implementare, all'interno di un widget python script di Orange, un algoritmo di Ottimizzazione Bayesiana tramite la libreria scritta in pure-python bayesian-optimization [1]. Questo approccio per individuare gli iperparametri migliori è stato preferito rispetto alla tradizionale Grid Search per via della sua maggiore efficienza computazionale, specialmente quando si lavora con spazi di iperparametri di grandi dimensioni. Mentre la Grid Search valuta esaustivamente tutte le possibili combinazioni di iperparametri in un dato insieme discreto, l'Ottimizzazione Bayesiana adotta un approccio iterativo che permette di identificare rapidamente le configurazioni ottimali, riducendo significativamente il tempo di calcolo.
L'Ottimizzazione Bayesiana si basa sull'idea di costruire un modello probabilistico del problema di ottimizzazione, chiamato anche modello surrogato, noto come processo gaussiano (GP). Questo modello viene utilizzato per stimare l'obiettivo in punti non ancora valutati e per determinare in modo strategico dove concentrare la ricerca degli iperparametri.
Nel nostro caso, l'obiettivo da massimizzare è l'accuratezza media calcolata attraverso una 5-fold cross-validation. Questo metodo è stato scelto in quanto il dataset è stato preventivamente bilanciato, rendendo l'accuratezza una metrica appropriata per valutare le prestazioni. Oltre all'accuratezza, si è tenuto conto in seguito anche delle metriche aggiuntive come l'F1-score e l'Area Under the ROC Curve (AUC-ROC), per garantire una valutazione completa delle prestazioni.
Sono stati ottimizzati tre modelli. Il primo è il Support Vector Classifier (SVC) con kernel RBF, per il quale gli iperparametri ottimizzati sono stati il parametro di regolarizzazione *C* e il parametro γ, che determina la larghezza del kernel. Il secondo modello è una rete neurale, per la quale gli iperparametri ottimizzati comprendevano il termine di regolarizzazione α e le dimensioni dei tre strati nascosti, consentendo di valutare configurazioni di rete con diversa complessità. Infine, per l'albero decisionale sono stati ottimizzati i parametri `min_samples_leaf`, `max_depth` e `min_samples_split`. La procedura per ciascun modello è stata strutturata con la definizione di una funzione obiettivo specifica, che calcola l'accuratezza media su una 5-fold cross-validation, e l'impostazione dei limiti per l'esplorazione di ciascun iperparametro. Successivamente, è stato utilizzato l'approccio iterativo dell'Ottimizzazione Bayesiana, che ha valutato 10 configurazioni iniziali casuali di iperparametri e iterato per ulteriori 25 step.

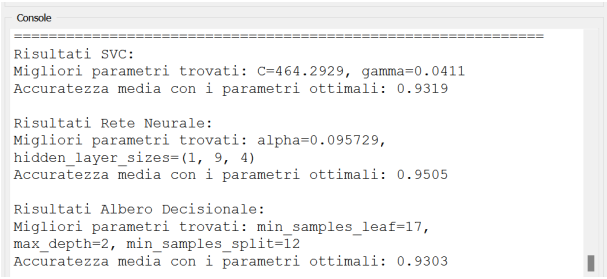

Fig. 4: Bayesian Optimization output for 1st Approach

Nel secondo approccio abbiamo voluto implementare una 5-fold Cross Validation eseguita contemporaneamente all'under-sampling del dataset, dove gli iperparametri di ciascun modello sono stati lasciati pari ai valori standard forniti dai widget di Orange. Per effettuare ciò abbiamo utilizzato a valle del dataset un widget Data Sampler impostato per generare 5 fold. Nei dataset sbilanciati il processo di cross validation viene quindi combinato con l'undersampling, dove i fold di training vengono sottoposti ad undersampling mentre il fold di test rimane sbilanciato ad ogni iterazione. Tramite i widget messi a disposizione dal software non era possibile però condurre le diverse iterazioni della CV in modo completamente automatizzato, ma solo una. Per tale ragione, al fine di ottenere risultati statisticamente affidabili, abbiamo effettuato manualmente cinque iterazioni, tramite un resampling dal widget Data Sampler Cross Validation, ed infine abbiamo calcolanto la media di tutte le metriche di prestazione ottenute ad ogni resample. Questo ci ha permesso di ottenere una valutazione quanto più oggettiva dei risultati di ogni modello ed indipendente da come vengono creati i fold di training e di test.

ITERAZIONE #1							
MODEL	AUC	CA	F1	PREC	RECALL	MCC	SPEC
SVM (1)	0,951	0,967	0,981	0,998	0,967	0,166	0,758
Random Forest	0,973	0,987	0,992	0,998	0,987	0,313	0,909
Neural Network	0,965	0,964	0,980	0,998	0,964	0,187	0,889

ITERAZIONE #2							
MODEL	AUC	CA	F1	PREC	RECALL	MCC	SPEC
SVM (1)	0,964	0,977	0,987	0,998	0,977	0,215	0,808
Random Forest	0,966	0,990	0,993	0,998	0,990	0,316	0,818
Neural Network	0,953	0,965	0,981	0,998	0,965	0,177	0,829

ITERAZIONE #3							
MODEL	AUC	CA	F1	PREC	RECALL	MCC	SPEC
SVM (1)	0,980	0,978	0,988	0,998	0,978	0,241	0,889
Random Forest	0,980	0,986	0,992	0,998	0,986	0,298	0,889
Neural Network	0,975	0,965	0,980	0,998	0,965	0,198	0,929

ITERAZIONE #4							
MODEL	AUC	CA	F1	PREC	RECALL	MCC	SPEC
SVM (1)	0,960	0,969	0,983	0,998	0,969	0,160	0,708
Random Forest	0,984	0,990	0,994	0,998	0,990	0,357	0,909
Neural Network	0,974	0,937	0,966	0,998	0,937	0,146	0,929

ITERAZIONE #5							
MODEL	AUC	CA	F1	PREC	RECALL	MCC	SPEC
SVM (1)	0,990	0,989	0,993	0,998	0,989	0,286	0,748
Random Forest	0,986	0,990	0,993	0,998	0,990	0,348	0,919
Neural Network	0,972	0,970	0,983	0,998	0,970	0,210	0,909

Fig. 5: Iterations 5-Fold CV for 2nd Approach

MEDIA MISURE DI PRESTAZIONE							
MODEL	AUC	CA	F1	PREC	RECALL	MCC	SPEC
SVM (1)	0,969	0,976	0,986	0,998	0,976	0,214	0,782
Random Forest	0,978	0,989	0,993	0,998	0,989	0,327	0,889
Neural Network	0,968	0,960	0,978	0,998	0,960	0,183	0,897

Fig. 6: Final metrics for 2nd Approach

Sulla base delle misure di prestazione, è possibile fare alcune osservazioni sui tre modelli di machine learning analizzati (SVM, Random Forest, e Neural Network):

1) Random Forest si distingue come il modello con le prestazioni complessivamente migliori. Ha il valore più alto per metriche cruciali come l'AUC (0.978), l'accuratezza (CA, 0.989), l'F1 score (0.993), e il Matthews Correlation Coefficient (MCC, 0.327), che misura la qualità delle predizioni bilanciando veri positivi e veri negativi. Anche la specificità (Spec, 0.889) e il recall (0.998) sono superiori rispetto agli altri modelli, dimostrando un equilibrio tra sensibilità e capacità di evitare falsi positivi.
2) SVM (Support Vector Machine) offre prestazioni buone, ma leggermente inferiori al Random Forest. L'accuratezza (CA, 0.976), l'F1 score (0.986) e l'AUC (0.969) indicano un modello solido, ma l'MCC (0.214) suggerisce che il modello potrebbe avere difficoltà a catturare correttamente la relazione tra le classi. La specificità (Spec, 0.782) è particolarmente bassa rispetto agli altri modelli, indicando una maggiore propensione ai falsi positivi.
3) La rete neurale presenta le prestazioni più basse tra i tre modelli in termini di accuratezza (CA, 0.960), F1 score (0.978) e MCC (0.183). Nonostante il recall sia elevato (0.998), la specificità (Spec, 0.897) è leggermente migliore rispetto a quella del SVM ma comunque inferiore al Random Forest. Questo suggerisce che il modello potrebbe privilegiare la cattura dei veri positivi a discapito della precisione nelle altre metriche.

VII. Analisi e validazione dei Modelli

Per quanto riguarda il primo approccio, si osserva che i risultati della metrica di accuratezza, calcolati in funzione degli stessi iperparametri ottimizzati tramite il widget utilizzato per l'Ottimizzazione Bayesiana, sono in linea con quelli ottenuti tramite il widget Test and Score, confermando così la robustezza delle procedure adottate. I lievi scostamenti osservati possono essere attribuiti a fattori numerici, come differenze nei metodi di inizializzazione dei pesi nella rete neurale e le modalità di generazione e suddivisione dei dati nei fold nella cross-validation. Questi elementi, pur influendo marginalmente, non alterano significativamente il trend generale delle prestazioni.

La rete neurale si distingue per prestazioni migliori rispetto agli altri modelli nelle metriche di accuratezza (CA), F1-score e Specificity (Spec). Ciò evidenzia una capacità superiore del modello nell'identificare correttamente sia i True Positives che i True Negatives, rendendolo particolarmente adatto al problema in analisi.

Infine, possiamo affermare che la consistenza tra le metriche restituite dai due widget rafforza la validità del processo di valutazione adottato. Questa coerenza implica che i risultati non dipendono da una specifica configurazione, ma rappresentano un comportamento stabile dei modelli.

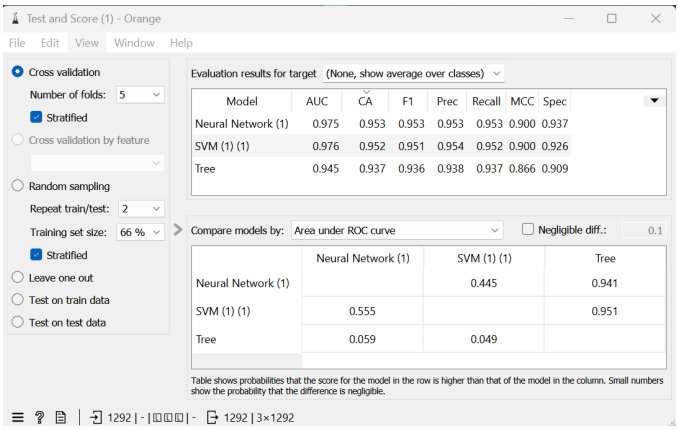

Fig. 7: Test and Score widget for 1st Approach

Per quanto riguarda il secondo approccio abbiamo appurato che il modello ensemble di Random Forest risulta più equilibrato e performante (Fig.6), adatto a un contesto in cui è necessario massimizzare sia la sensibilità sia la specificità, mantenendo una solida accuratezza complessiva. Il modello SVM può essere utile in scenari in cui la semplicità del modello e la velocità di predizione siano prioritarie, ma richiederebbe ottimizzazioni per migliorare specificità e MCC. Mentre la Neural Network, pur mostrando un buon recall, necessita probabilmente di un'ulteriore fase di tuning per competere con le prestazioni del Random Forest. In conclusione, Random Forest sembra essere la scelta più robusta per l'approccio considerato. Risulta evidente che le performance, generalmente scarse, della metrica MCC portino la scelta a ricadere sul primo approccio, ciò nonostante gli autori di questo report, considerati gli studi preliminari effettuati da esperti data scientist su questo dataset [2], suggeriscono di valutare anche l'utilità del secondo approccio, onde evitare rischi presentabili nel primo caso, quale ad esempio overfitting o bias dovuti ad un test dei modelli su un fold bilanciato che non rispecchia la reale distribuzione della classe target.

VIII. Conclusioni

I risultati ottenuti dal secondo approccio evidenziano che il modello Random Forest si distingue per prestazioni complessive migliori, rendendolo adatto a contesti in cui la precisione e la rilevazione tempestiva delle frodi sono prioritarie. Tuttavia, sono emerse opportunità di miglioramento, in particolare nel confronto tra approcci basati su undersampling: uno eseguito prima dell'addestramento e uno integrato nella procedura di cross-validation.
Una direzione promettente per sviluppi futuri include l'applicazione dell'Ottimizzazione Bayesiana anche nell'approccio di undersampling dinamico durante la cross-validation. Questo permetterebbe di valutare l'impatto di diverse configurazioni di iperparametri in un contesto più vicino alla realtà operativa, migliorando la generalizzazione dei modelli e riducendo il rischio di overfitting. Infine, ulteriori ricerche potrebbero esplorare l'uso di tecniche di oversampling per la classe minoritaria, come la tecnica SMOTE e l'implementazione di modelli ensemble avanzati per affrontare al meglio la complessità del problema.

References

[1] F. Nogueira, *Bayesian Optimization: Open source constrained global optimization tool for Python*, 2014. [Online]. Available: https://bayesian-optimization.github.io/BayesianOptimization/2.0.3/

[2] J. Bachmann, *Credit Fraud - Dealing with Imbalanced Datasets*, Kaggle Notebook, 2021. [Online]. Available: https://www.kaggle.com/code/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets