UNIVERISTÀ DI PISA
SCUOLA DI INGEGNERIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

BoCSoR, SHAP e Permutation Importance: analisi comparativa di metodi di misurazione della importanza delle caratteristiche per il riconoscimento di movimenti immaginati da dati EEG

12 dicembre 2024

Candidato

Elia Tonci

Relatori

Ing. Antonio Luca Alfeo

Prof. Mario G.C.A. Cimino

Sommario

Abstract

L'analisi dell'importanza delle caratteristiche in modelli di machine learning (ML) è fondamentale per la comprensione dei processi decisionali dei modelli, specialmente quando questi vengono applicati a dati complessi e dinamici, come quelli provenienti da sistemi di produzione intelligente. Con l'espansione delle applicazioni di ML in contesti critici, come la medicina per esempio, diventa essenziale fornire spiegazioni comprensibili per gli utenti non esperti, senza compromettere l'affidabilità del modello. Le tecniche di intelligenza artificiale spiegabile (XAI) si sono rivelate uno strumento potente per raggiungere questo obiettivo, ma molte di queste tecniche sono caratterizzate da un elevato costo computazionale, che le rende poco pratiche per applicazioni in tempo reale o su larga scala.

Questo lavoro si concentra sull'applicazione di tecniche XAI per l'analisi dell'importanza delle caratteristiche in modelli di classificazione applicati a dataset di benchmark e dati reali provenienti da ambienti di produzione intelligente. L'approccio proposto si distingue per l'uso di metodi ibridi, che combinano le soluzioni tradizionali con nuove strategie più efficienti dal punto di vista computazionale. In particolare, viene esplorato come l'influenza di piccole modifiche nelle caratteristiche possa alterare i risultati di classificazione, e come l'integrazione della conoscenza del dominio possa aiutare a ottenere spiegazioni più robuste, riducendo al contempo il rischio di introdurre bias nei modelli.

I risultati ottenuti mostrano che i metodi ibridi proposti riescono a mantenere un buon equilibrio tra l'affidabilità delle spiegazioni e l'efficienza computazionale. Inoltre, viene evidenziato come l'approccio possa essere applicato con successo a scenari pratici, come il monitoraggio delle performance di sistemi intelligenti in tempo reale, migliorando la comprensione delle decisioni del modello e facilitando l'intervento umano quando necessario. I confronti tra i metodi tradizionali e quelli ibridi indicano che le soluzioni proposte non solo sono più veloci, ma offrono anche un livello di comprensione delle decisioni del modello equivalente, se non superiore, a quelle ottenute con tecniche più costose. Questo studio contribuisce a rendere l'uso delle tecniche XAI più accessibile ed efficiente per applicazioni industriali e altre aree dove le risorse computazionali sono limitate.

1. Introduzione

L'Intelligenza Artificiale (AI) è emersa come uno strumento potente per l'analisi dei dati fisiologici. Questo tipo di dati, che include segnali come elettrocardiogrammi (ECG), elettroencefalogrammi (EEG), variazioni della frequenza cardiaca, e altre misurazioni biometrico-fisiologiche, è caratterizzato da complessità e multidimensionalità. L'AI si è dimostrata particolarmente utile nel gestire tali complessità, fornendo modelli in grado di rilevare anomalie, prevedere stati clinici o monitorare condizioni di salute in tempo reale.

Tuttavia, l'applicazione dell'AI in ambiti sensibili come la medicina e il monitoraggio fisiologico presenta un problema critico: la fiducia degli esperti di dominio nei risultati generati dai modelli AI. In contesti clinici e sanitari, dove decisioni diagnostiche o terapeutiche possono avere conseguenze significative per la salute e la vita dei pazienti, ogni risultato prodotto dall'AI deve essere non solo accurato, ma anche comprensibile e verificabile dagli operatori umani. Gli esperti, come medici e ricercatori, che utilizzano questi strumenti per supportare il processo decisionale, sono restii ad accettare predizioni o raccomandazioni che non possono essere pienamente comprese o giustificate. Questo scetticismo è particolarmente pronunciato in quanto gli errori, anche rari, possono avere impatti devastanti, erodendo ulteriormente la fiducia nella tecnologia.

Il problema è ulteriormente aggravato dal fatto che molti modelli AI, specialmente quelli basati su reti neurali profonde (Deep Learning), funzionano come "scatole nere". Questo significa che, pur essendo in grado di produrre predizioni accurate, non forniscono una spiegazione chiara e intuitiva del ragionamento che ha portato a quelle conclusioni. Per esempio, un modello di AI può classificare un ECG come anomalo o predire un rischio elevato di aritmia cardiaca, ma spesso non è in grado di specificare quali caratteristiche del segnale o quali pattern hanno contribuito a tale risultato in modo interpretabile.

Questo limite ha implicazioni profonde:

1. Difficoltà di validazione clinica: Senza una spiegazione chiara, è difficile per gli esperti validare se le predizioni sono coerenti con i principi medici o con le conoscenze cliniche esistenti. La fiducia è essenziale in medicina, e senza spiegazioni comprensibili, i modelli AI rischiano di essere percepiti come inaffidabili o inutilizzabili.
2. Responsabilità e decisioni etiche: In ambiti sensibili come la salute, le decisioni devono essere giustificabili. Un medico non può basarsi esclusivamente su un risultato generato da un sistema opaco senza la capacità di spiegare ai pazienti o alle autorità come è stata raggiunta quella conclusione. La mancanza di spiegazioni comprensibili rende difficile anche assegnare la responsabilità in caso di errore.
3. Accettazione e adozione limitata: Anche i modelli AI più avanzati e accurati rischiano di non essere adottati se gli esperti non possono fidarsi di loro. Questo crea una barriera significativa per l'integrazione della tecnologia nei flussi di lavoro clinici.

L'AI ha il potenziale di rivoluzionare la medicina, migliorando l'accuratezza diagnostica, la velocità di analisi e l'efficienza complessiva. Tuttavia, senza meccanismi che permettano di spiegare i risultati generati, la tecnologia rischia di rimanere inutilizzata o utilizzata in modo limitato. La fiducia degli esperti dipende dalla capacità di rendere il funzionamento

dei modelli trasparente e interpretabile, garantendo che ogni predizione possa essere giustificata sulla base di evidenze comprensibili e verificabili.

Questa esigenza di fiducia e trasparenza ha spinto la comunità scientifica verso lo sviluppo di tecnologie di Intelligenza Artificiale Spiegabile (XAI). Questi approcci mirano a fornire spiegazioni umanamente comprensibili per i risultati prodotti dai modelli AI, riducendo il divario tra le capacità predittive della tecnologia e le esigenze degli esperti di dominio.

Gli approcci XAI possono fornire alcune spiegazioni per i risultati, riducendo i costi e aumentando l'efficienza della produzione. Seguendo le definizioni di interpretabilità e fedeltà, in relazione ad una spiegazione, capiamo che le spiegazioni interpretabili, tipicamente, non sono fedeli e viceversa. La spiegaziona usata per essere comprensibile a chi utilizza l'AI, deve essere il più possibile interpretabile. Vengono utilizzati due tipi di spiegazioni principali: le spiegazioni basate sull'attribuzione, come l'importanza delle caratteristiche che valutano l'importanza di parti dell'input per una data classificazione, sono una tipologia molto costosa computazionalmente, e le spiegazioni basate sull'istanza, come quelle basate su controfattuali che espongono somiglianze e differenze tra un'istanza classificata dal modello di AI e istanze simili di una classe diversa, questa tipologia può spiegare un solo risultato e non tutto il modello. Per evitare i loro svantaggi posso combinarli, per fornire nuoi approcci XAI. Un problema riguardo i nuovi metodi XAI è la loro validazione, le ipotesi alla base delle spiegazioni potrebbero non essere verificate in nessuno scenario reale. Ad esempio, c'è rischio che il metodo XAI sovrastimi o sottostimi l'importanza di una caratteristica, poiché le valutazioni a priori, di una specifica caratteristica, potrebbero non essere esaustive. Vengono usati due approcci per affrontare questo problema, la generazione di dataset con importanza relativa delle caratteristiche nota oppure una spiegazione della verità di fondo tramite classificatori trasparenti. Entrambi, però, potrebbero iniettare un certo bias nei dati e nelle spiegazioni generate, riducendo affidabilità ed efficacia. Questo problema viene affrontato utilizzando un dataset reale in cui l'importanza prevista di ciascuna caratteristica nei dati viene fornita dagli esperti di dominio e utilizzata per validare una misura di importanza delle caratteristiche che è indifferente rispetto al modello globale e basata sui controfattuali.

1.1 Permutation Importance e SHAP

Le misure di importanza come Permutation Importance e SHarpley Additive exPlanation (SHAP), sono tra gli strumenti di spiegazione più usati. Il primo misura l'importanza di ciascuna caratteristica per un modello AI addestrato. Questo processo interrompe la relazione tra una caratteristica e la classe target. Il secondo, considerato uno standard di riferimento tra gli approcci all'importanza delle caratteristiche, anche se molto costoso, valuta l'importanza di una caratteristica per la classificazione misurando il contributo marginale medio di quella caratteristica su tutti i possibili sottoinsiemi di caratteristiche. L'utilizzo di spiegazioni controfattuali porta ad un costo ridotto, tuttavia non forniscono informazioni sul ragionamento del modello AI.

1.2 BoCSoR

Per ottenere la produzione migliore, come già detto, combiniamo le due tipologie di spiegazioni. La misura di importanza delle caratteristiche impiegata in questo studio, ovvero BoCSoR, appartiene a un nuovo filone della letteratura XAI, in cui vengono combinate diverse strategie di spiegazione per affrontare le loro limitazioni. L'idea principale dietro BoCSoR, Boundary Crossing Solo Ratio, è valutare l'importanza di una caratteristica considerando la frequenza con cui i campioni vicini al confine decisionale del modello risultano in un risultato di classificazione diverso se il valore di quella specifica caratteristica viene sostituito con quello del corrispondente campione controfattuale. Infine, BoCSoR valuta l'importanza di una caratteristica considerando la frequenza con cui i cambiamenti di essa, da soli, risultano nell'attraversamento del confine decisionale del modello, considerando i campioni vicini ad esso. BoCSoR risulta in una complessità temporale caratterizzata da una crescita lineare rispetto al numero di caratteristiche (F) e una crescita quadratica rispetto al numero di campioni (N), ovvero O(N^2 + N · F · log(N)), inferiore rispetto a SHAP.

2. Related works

Si mostrano ora due studi affrontano lo stesso problema affrontato in questa tesi, però con il punto di vista degli anni passati. Ora vogliamo superare i risultati ottenuti da questi lavori.

Peeking inside the black-box: a survey on explainable artificial intelligence. - Adadi A., Berrada M. – 2018

L'artcolo afforonta il fatto che la crescente complessità dei modelli di intelligenza artificiale, come le reti neurali profonde, li rende "black-box", difficili da interpretare. Tuttavia, la spiegabilità è fondamentale per favorire l'accettazione, la fiducia e l'adozione di tali modelli. Viene posto come obiettivo di spiegare le tecniche XAI e di discutere le loro applicazioni e le loro sfide.

Viene introdotta l'importanza della spiegabilità, soprattutto in settori come la medicina e nella giustizia, dove le decisioni hanno consequenze significative. Si parla anche del fatto che un sistema AI deve essere equo e non discriminatorio.

Si mettono quindi a confronto due tipologie di teciche: intrinseche e post-hoc. Le prime riguardano modelli progettati per essere interpretabili, le seconde riguardano metodi applicati a modelli complessi per spiegarne il comportamento. Fanno parte del secondo insieme SHAP e Permutation Importance che sono state prese in considerazione anche nel nostro studio.

La XAI deve affrontare il fatto che i modelli più interpretabili sono spesso meno accurati rispetto ai modelli black-box, che non esiste un consenso su come misurare o valutare la qualità delle spiegazioni e che le tecniche devono essere adattabili a vari domini e modelli.

Si arriva alla conclusione che la XAI è essenziale per aumentare la fiducia nei sistemi di IA. Sebbene siano stati compiuti progressi significativi, rimangono aperte sfide tecniche e filosofiche. È necessaria una maggiore collaborazione tra ricerca e applicazioni pratiche per sviluppare metodi di spiegabilità robusti e standardizzati. Infatti molti metodi sono ancora troppo costosi a livello computazionale oppure sono poco accurati per aumentare l'interpretabilità.

A unified approach to interpreting model predictions. - Lundberg S.M., Lee S.I. – 2017

Viene presentato un framework chiamato SHAP (SHapley Additive exPlanations) per interpretare le predizioni di modelli complessi, unificando diverse tecniche di attribuzione delle feature basate su valori di Shapley.

Il problema da risolvere si basa sul fatto che modelli complessi come reti neurali ed ensemble methods offrono alta accuratezza, ma spesso sono difficili da interpretare. Questo crea una tensione tra accuratezza e interpretabilità, fondamentale in applicazioni critiche. Sono state proposte diverse tecniche per spiegare i modelli, ma mancano spesso di un fondamento teorico comune e non garantiscono proprietà desiderabili come accuratezza locale e consistenza.

La proposta per risolvere questo problema è il framework SHAP. Esso si basa sui valori di Shapley dalla teoria dei giochi, che calcolano l'importanza di ogni feature in base al suo

contributo marginale. SHAP unifica sei metodi già esistenti, inclusi LIME, DeepLIFT e layer-wise relevance propagation.

Si definisce una classe di metodi di attribuzione delle feature additive con tre proprietà chiave: accuratezza locale, questa proprietà garantisce che la somma delle attribuzioni delle feature (i valori di Shapley) corrisponda esattamente alla predizione del modello per un'istanza specifica, mancanza di impatto delle feature assenti e consistenza, ovvero che se si modifica un modello in modo che aumenti la dipendenza della predizione da una feature, il valore attribuito a quella feature deve aumentare o, al massimo, rimanere invariato.

Grazie a SHAP si dimostra che c'è un'unica soluzione nella classe, che coincide con i valori di Shapley.

SHAP viene valutato su benchmark e confrontato con altri metodi di interpretabilità. I risultati mostrano un miglioramento nella coerenza delle spiegazioni. I punti negativi di SHAP sono principlamente due: elevati costi computazionali; infatti, esso è un processo esponenziale che diventa rapidamente impraticabile per modelli complessi o dataset di grandi dimensioni, e la complessa implementazione del metodo.

3. Design e implementazione

In questa ricerca utilizziamo un dataset reale, vengono messi a confronto più metodi, tra cui BoCSoR, nel processare i vari dati presenti in esso. Si calcola l'importanza delle caratteristiche del dataset e quanto tempo ogni metodo impiega a generarle. In questo modo possiamo fare un confronto sia sull'affidabilità del metodo che stiamo studiando rispetto agli altri, ormai considerati standard, sia sulle differenze riguardo il costo computazionale. Per avere un effettivo confronto tra le importanze calcolate utilizzo la Pearson correlation e il p-value.

I metodi vengono ripetuti più volte per non avere un solo risultato da osservare, così da poter stimare l'effettiva efficacia di BoCSoR, con l'obiettivo che quest'ultimo, combinando diverse strategie, porti a costi computazionali ridotti e a risultati comunque affidabili.

3.1 Pearson Correlation e p-value

Come abbiamo detto, per valutare BoCSor non basta fare il confronto con gli altri metodi di misura di importanza riguardo il tempo necessario per ottenere i risultati, ma devo anche confornare l'importanza, relativa ad ogni dato, ottenuta. Riguardo quest'ultimo fattore, il metodo utilizzato più affidabile e l'Intrinsic, per questo devo misurare la fedeltà che ogni altro metodo ha con essso e poi confronto ciò che ottengo per ogni metodo. Per misurare la fedeltà di un metodo rispetto ad un altro utilizzo due valori precisi: la Pearson Correlation e il p-value.

Il coefficiente di correlazione di Pearson misura la relazione lineare tra due set di dati. Rigorosamente parlando, la correlazione di Pearson richiede che ciascun set di dati sia distribuito normalmente. Varia tra -1 e +1, con 0 che implica assenza di correlazione. Correlazioni di -1 o +1 implicano una relazione lineare perfetta. Dati x e y set di dati della stessa dimensione: correlazioni positive implicano che, all'aumentare di x, aumenta anche y, mentre correlazioni negative implicano che, all'aumentare di x, y diminuisce. Se il valore ottenuto è:

- maggiore di 0.7, o minore di -0.7, abbiamo una correlazione forte
- compreso tra 0.3 e 0.7, o compreso tra -0.3 e -0.7, abbiamo una correlazione moderata
- compreso tra -0.3 e 0.3 abbiamo una correlazione debole.

Il p-value indica approssimativamente la probabilità che un sistema non correlato produca set di dati con una correlazione di Pearson almeno estrema quanto quella calcolata dai set di dati in esame. I p-value non sono del tutto affidabili, ma sono probabilmente ragionevoli per set di dati più grandi, di circa 500 elementi o più. Il p-value è considerato basso se è minore di 0.05, significa che il risultato osservato non è così raro e sembra più in linea con quello che ci aspettiamo normalmente. Altrimenti il p-value è considerato alto, indica che il risultato osservato è abbastanza raro rispetto a quello che ci aspetteremmo di solito. Per vedere questa differenza con un esempio più familiare considero il lancio di una moneta per dieci volte:

- un p-value basso corrisponde ad ottenere 5 volte testa e 5 volte croce, ovvero ciò che ci aspettiamo

- un p-value alto corrisponde ad ottenere 9 volte testa ed una sola volta croce, un risultato inaspettato

Grazie a questi valori sono in grado di controllare l'affidabilità dei risultati ottenuti con BoCSoR, valutandone la fedelta ad Intrinsic, anche confrontando questi risultati con gli altri ottenuti con altri metodi.

3.2 Librerie utilizzate

Per calcolare le importanze, i tempi e i parametri descritti sopra utilizziamo come linguaggio di programmazione il python. Questo, grazie alla vasta gamma di librerie messe a disposizione e all'intuitiva sintassi, è tra i linguaggi più usati a livello mondiale per l'analisi di dati scientifici.

In particolare python mette a disposizione una libreria, chiamata Shap, che ci permette di implementare il metodo SHAP, dato che fornisce strumenti per interpretare predizioni di modelli di machine learning, basandosi sui valori di Shapley della teoria dei giochi.

Altre librerie importanti per i calcoli sono: NumPy (Numerical Python), fondamentale per il calcolo scientifico, permette di gestire array multidimensionali e di eseguire operazioni matematiche efficenti su di essi, Sklearn (Scikit-learn), è una libreria specidica per il machine learning, fornisce algoritmi e strumenti per pre-processing, modellazione e validazione, Pandas, ci permette di manipolare e analizzare dati strutturati, come tabelle e file CSV, e SciPy, altra libreria per il calcolo scientifico avanzato, in particolare il sottopacchetto Stats implementa il metodo Pearsonr che ci permette di calcolare la Pearson Correlation e il p-value.

3.3 Classi implementate

Al fine di calcolare le importanze dei metodi e di misurare i tempi necessari per farlo, implementiamo delle classi che ci permettono di creare un codice ben strutturato e semplice da capire.

- **FeatureImportanceClassifier**: questa classe ha come scopo principale il calcolo effettivo delle importanze, utilizzando i diversi modelli che gli vengono dati in input e i diversi dataset che vogliamo processare. Calcola le importanze utilizando metodi prestabiliti, che però possono essere modificati, aggiungendone altri. Qui vengono anche create le tabelle che andranno poi a contenere i dati misurati.

- **DataLoader**: questa classe carica il dataset nel codice e lo rende processabile dai vari metodi. Fornisce le divisioni necessarie, ovvero le features, le labels, le numerical_features, le categorical_features e i column_names

- **BoCSoR**: implementa tutti i metodi ibridi di BoCSoR (BoCSoR Distance Original, BoCSoR Distance Weighted, BoCSoR Random Original, BoCSoR Random Weighted).

- **CounterfactualExplainerByProximity**: è progettata per spiegare e generare controfattuali (cioè esempi ipotetici che potrebbero cambiare la previsione di un modello) per un dato campione in un contesto di machine learning, con un focus sul modello basato sulla vicinanza

3.4 Casi d'uso

Dal software si possono riconoscere due casi d'uso, partendo dal fatto che, come risultato finale, otteniamo due tabelle: una contenente le importanze calcolate e una contenente tempo di calcolo, Pearson Correlation e p-value. Il caso d'uso principale e il confronto diretto che possiamo fare tra 3 più metodi utilizzati. La seconda tabella ci permette infatti di capire se un metodo impiega meno tempo di un altro ad eseguire i vari calcoli e se la sua affidabilità è più alta oppure no. Avere i dati calcolati su più ripetizioni ci permette di fare la media su di essi e ottenere un valore più generale, che aumenta di precisione all'aumentare delle ripetizioni svolte. Grazie a tutto questo possiamo valutare se un qualunque nuovo metodo, da testare, è affidabile oppure no e valutare quest'ultima caratteristica a fronte del tempo che impiega a processare il dataset.

Un secondo caso d'uso è semplicemente il calcolo delle importanze dei vari dati del dataset in input. La prima tabella è formattata in questo modo: sulle righe sono presenti gli indici dei dati, sulle colonne sono presenti i metodi utilizzati per ogni ripetizione; quindi, ogni metodo comparirà un numero di volte equivalente al numero di ripetizioni. Nei vari domini della tabella saranno presenti le importanze ralative al dato e al metodo in causa. Questo ci permette quindi di valutare singolarmente ogni dato grazie all'importanza ottenuta processando il dataset con un determinato metodo.

4. Case study

Partendo dal dataset BCI_2, processo il dataset utilizzando diversi metodi, oltre ad Intrinsic: SHAP, Permutation, Dice e i 4 metodi relativi a BoCSoR, BoCSoR Distance Original, BoCSoR Random Weighted, BoCSoR Distance Weighted, BoCSoR Random Original. Dice non lo considero, le importanze calcolate con questo metodo vengono messe a zero manualmente.

Considero due modelli di machine learning: Decision Tree, più semplice e generico, e Random Forest, più preciso ma anche più complesso. Consideriamo che un Decision Tree è un singolo albero di decisione che divide iterativamente i dati in base a una serie di condizioni, cercando di massimizzare la separazione tra le classi. Una Random Forest è un insieme di Decision Tree, combinati per ottenere una previsione finale più robusta e accurata. Quest'ultimo è più accurato e i risultati sono meno influenzati da variazioni nei dati di addestramento. Tutto questo a discapito della semplicità del Decision Tree e con un aumento significativo del tempo di calcolo. Una Random Forest crea una collezione di Decision Tree addestrati su diversi sottoinsiemi casuali del dataset. Per ogni albero, a ogni nodo viene considerato solo un sottoinsieme casuale di variabili per scegliere la migliore divisione.

Voglio ottenere un confronto anche per diversi utilizzi di Random Forest, ovvero capire come cambiano i risultati a seconda di quanti alberi prendo in considerazione. Per ottenere ciò creo quattro tabelle diverse, la prima utilizzando una foresta di 5 alberi, la seconda utilizzando una foresta di 10 alberi, la terza utilizzando una foresta di 15 alberi e la quarta utilizzando una foresta di 50 alberi. L'ultima sarà quella più costosa in termini computazionali ma sarà anche quella più precisa.

Effettuerò questo processo per cinque volte ottenendo così 5 fold. Meglio che averne solo una per poter fare una valutazione più accurata.

Per ottenere le importanze carico in un array il dataset elaborato, vengono divise le features dalle labels, creo due array specifici che conterrano tutte le importanze calcolate in ogni ripetizione e tutti i tempi di computazione misurati. Per ogni ripetizione:

- creo due array che conterranno le importanze e i tempi misurati, relativi a tale ripetizione.
- per ogni metodo mi calcolo le importanze e il tempo necessario per calcolarle e inserisco questi dati nei relativi array.
- concateno i risultati ottenuti nei due array creati inizialmente, cosicchè alla fine di tutte le ripetizioni, questi contangano effettivamente tutte le importanze e i tempi misurati.

Ogni fold, quindi, contiene l'importanza misurata da ogni metodo per ogni dato. Ottengo una tabella che ha come righe i dati del dataset e come colonne i metodi di ogni fold (es. Fold 1 SHAP, Fold 2 SHAP, Fold 3 SHAP etc.).

La prima cosa da fare e creare un array di valori per ogni metodo di ogni fold, per un totale di 35 array (5 fold * 7 metodi). Ora, per ogni fold, mi calcolo la correlazione e il p-value tra Intrinsic e ognuno degli altri sei metodi. I valori li calcolo tramite la funzione (appartenente alla libreria *scipy.stats*):

pearson_coef, p_value = pearsonr(fold_intrinsic, metodo_da_confrontare)

Equazione 1 Pearson Correlation Coefficent e p-value

Questa funzione prende in input due vettori della stessa lunghezza. Nel nostro caso abbiamo come primo parametro le importanze calcolate con il metodo Intrinsic, relative ad una precisa fold, ovvero una precisa ripetizione del calcolo. Come secondo parametro abbiamo le importanze calcolate con un qualunque altro metodo tra quelli stabiliti a priori, sempre relative alla stessa fold considerata dal metodo Intrinsic. Quindi questa funzione dovrà essere ripetuta un numero di volte pari al numero di metodi che devono essere confrontati con Intrinsic. La funzione in questione è molto utile e comoda, datoc che ci restituisce direttamente i due valori che necessitiamo, senza dover utilizzare la definizione formale del calcolo della correlazione di Pearson e del p-value, decisamente più complessa e costosa.

Ora che ho calcolato i due valori per ogni metodo di ogni fold, posso creare una tabella per confrontarli e capire se i dati di BoCSoR sono affidabili. Nella tabella ho due righe per ogni metodo, una contiene il tempo di computazione e una il coefficiente di correlazione e il p-value, calcolati come detto prcedentemente. Invece ogni colonna indica una fold diversa, ovvero una ripetizione diversa.

Infine, calcolo la media dei valori di ogni riga e aggiungo una colonna che possa contenere questo nuovo valore. Calcolo la media per far in modo che il confronto sia più leggibile dall'occhio umano e che quindi i valori siano più interpretabili.

5. Risultati sperimentali

5.1 Metriche utilizzate

Per valutare i risultati ottenuti si utilizzano due metriche diverse per valutare due caratteristiche diverse. Le metriche sono il tempo necessario per calcolare le importanze e la Pearson Correlation legata al p-value. Entrambe sono fondamentali per avere un'analisi completa, in quanto dobbiamo capire se un metodo che ha tempi di calcolo molto bassi riesce ad avere anche una correlazione alta, e quindi una maggiore fedeltà con il metodo intrinsic, e viceversa, cioè capire se un metodo che restituisce valori di di correlazione alti e p-value bassi riesce a calcolare le importanze in tempi ristretti.

5.1.1 Tempo Computazionale

Il tempo visto nelle tabelle è calcolato in secondi. Viene indicato anche il tempo di calcolo necessario al metodo Intrinsic per calcolare le importanze, questo è per tutte le fold molto basso, essendo il metodo preso come riferimento per il calcolo della correlazione questo ci prova che è il metodo migliore. I tempi calcolati utilizzando il Decision Tree come modello saranno molto ridotti in generale; invece, utilizzando il modello Random Forest otteniamo, ovviamente tempi più lunghi, ma che ci fanno comprendere situazioni che il primo modello non mostra.

5.1.2 Pearson Correlation e p-value

Come abbiamo visto la Pearson Correlation è un valore che è compreso tra -1 e +1, dove questi ultimi due valori indicano massima fedeltà. Nel nostro studio considereremo una crrelazione buona quando il valore è superiore a 0,5 o inferiore a -0,5, in questo modo riusciamo a valutare due casi distinti.

Il p-value invece è un indice che viene considerato buono quando inferiore a 0,05. Nel nostro studio otteniamo più raramente valori che si avvicinano a questo numero, in generale sono tutti molto inferiori, raggiungendo anche l'odine di 10^-18 per esempio, quindi possiamo considerarlo meno fondamentale, ma comunque utile per vedere quali metodi effettivamente ci danno valori più alti.

Ovviamente non vengono riportati i valori prodotti con il metodo Intrinsic, dato che esso stesso è il valore preso come riferimento e quindi la Pearson Correlatione sarebbe sempre uguale a 1 e il p-value sempre uguale a 0.

5.1.3 Indice unificante

Partendo da questi tre valori possiamo provare a pensare ad un indice, compreso tra 0 e 1, che unifica l'importanza dei tre, dando un peso diverso a ciascuno. In questo modo potremmo avere una stima della qualità di un determinato metodo su un determinato modello.

Partendo dai tre valori calcolati mi vado a calcolare i corrispondenti valori normalizzati, ovvero compresi tra 0 e 1. In particolare, devo fare in modo che più i valori del tempo e del p-value sono bassi più aumenta l'indice, dato che questi valori più sono bassi e più aumenta la qualità del metodo. Invece per la Pearson Correlation devo semplicemente fare in modo che il suo valore passi da essere compreso tra -1 e 1 ad essere compreso tra 0 e 1.

$$TempoNorm = \frac{1}{1 + Tempo}$$

Equazione 2 Tempo Normalizzato

$$CorrelazioneNorm = \frac{1 + Correlazione}{2}$$

Equazione 3 Pearson Correlation Normalizzata

$$PvalueNorm = \frac{1}{1 + Pvalue}$$

Equazione 4 p-value Normalizzato

Ottenuti i nuovi valori devo assegnare ad ognuno un peso che avranno sull'indice, in maniera arbitraria.

- Il tempo avra come peso 0.3 dato che è importante, ma i metodi con tempi computazionali più brevi sono utili solo se non sacrificano la qualità del risultato. Quindi, il tempo è un fattore secondario rispetto alla qualità statistica del risultato.
- La Pearson Correlation avrà il peso maggiore, equivalente a 0.5, perché è un indicatore di quanto un metodo si avvicini a un altro metodo affidabile, Intrinsic nel nostro caso. Se il metodo è fortemente correlato con un metodo di riferimento, significa che è efficace nel rappresentare il fenomeno che stiamo cercando di modellare.
- Il p-value avra il peso minore, ovvero 0.2, esso è una misura statistica della significatività del risultato. Un p-value basso è desiderabile, ma non è sempre il fattore più determinante se la correlazione è forte.

Quindi l'indice finale si ottiene in questo modo:

$$Indice = TempoNorm \cdot 0.3 + CorrelazioneNorm \cdot 0.5 + PvalueNorm \cdot 0.2$$

Equazione 5 Indice

Più il valore di questo indice si avvicina ad 1 e maggiore è la qualità del metodo corrispondente. Se calcoliamo l'indice sul metodo Intrinsic, che è il metodo di riferimento per il calcolo della correlazione e del p-value, otteniamo un valore tendente a 1, dato che il tempo normalizzato non può essere 1 in quanto il tempo di calcolo non può essere 0.

5.2 Decision Tree

Inizialmente andiamo ad usare come modello Decision Tree, quindi andremo ad esplorare un solo albero decisionale. I tempi, generalmente, sono abbastanza ridotti data la bassa complessità del modello. Andiamo a valutare se utilizzare BoCSoR come metodo ibrido ci conviene rispetto ad utilizzare altri metodi standardizzati come SHAP e Permutation Importance.

Decision Tree						
Method	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Computational Time SHAP	268,8471	271,2831	262,1370	263,3667	264,4102	266,0088
Computational Time BoCSoR Distance Original	87,3367	86,0969	87,9991	80,5868	88,6519	86,1343
Computational Time BoCSoR Random Weighted	17,0924	17,7847	16,9955	16,8903	17,1686	17,1863
Computational Time BoCSoR Distance Weighted	57,3556	60,5806	59,5251	59,8281	59,0837	59,2746
Computational Time BoCSoR Random Original	17,2199	16,7838	16,7538	16,6395	16,6263	16,8047
Computational Time Permutation	107,3060	113,0362	119,0010	105,8703	105,7430	110,1913
Computational Time Intrinsic	2,3365*10^-5	2,5510*10^-5	3,5047*10^-5	4,0531*10^-5	2,4080*10^-5	2,9707*10^-5

Tabella 1 Tempi computazionali Decision Tree

Si può mostrare la differenza usando anche un grafico lineare, utilizzando la media calcolata dei vari tempi di tutti i metodi.

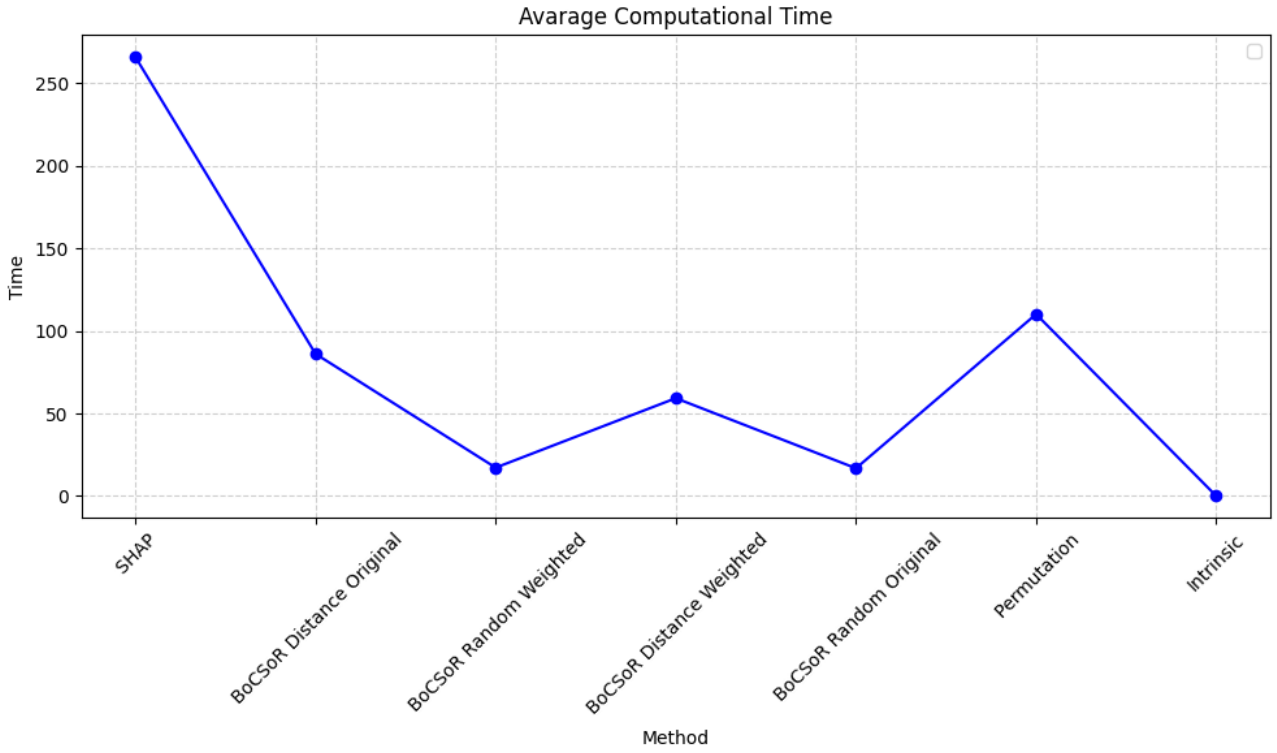

Figura 1 Grafico tempi computazionali Decision Tree

Sia dalla tabella sia dal grafico vediamo come SHAP è il metodo con tempi computazionali maggiori. Permutation Importance ha tempistiche inferiori, più vicine ai tempi dei vari metodi BoCSoR. Il metodo con tempo medio minore è BoCSoR Random Original. Possiamo vedere che, comunque, tutti i metodi BoCSoR hanno tempi ridotti e più convenienti, soprattutto rispetto a SHAP.

Ora possiamo valutare la correlazione con Intrinsic e il p-value corrispondente, così da poter valutare se il tempo richiesto da SHAP è giustificato da valori molto alti di quest'ultimi parametri.

Decision Tree						
Method	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Correlation (p-value) SHAP	0.5993 (3.14*10^-14)	0.6342 (3.23*10^-16)	0.6987 (1.21*10^-20)	0.7691 (4.73*10^-27)	0.6321 (4.35*10^-16)	0.6667 (6.43*10^-15)
Correlation (p-value) BoCSoR Distance Original	0.4163 (6.85*10^-7)	0.5052 (6.49*10^-10)	0.5935 (6.37*10^-14)	0.6638 (4.12*10^-18)	0.5161 (2.38*10^-10)	0.5390 (1.37*10^-7)
Correlation (p-value) BoCSoR Random Weighted	0.4748 (8.79*10^-9)	0.4464 (8.08*10^-8)	0.5784 (3.76*10^-13)	0.7026 (6.01*10^-21)	0.6137 (5.1*10^-15)	0.5632 (1.79*10^-8)
Correlation (p-value) BoCSoR Distance Weighted	0.1451 (0.0968)	0.5545 (5.24*10^-12)	0.2923 (6.69*10^-4)	0.3462 (4.76*10^-5)	0.5177 (2.05*10^-10)	0.3712 (1.95*10^-2)
Correlation (p-value) BoCSoR Random Original	0.5386 (2.7*10^-11)	0.6424 (1.01*10^-16)	0.5997 (2.98*10^-14)	0.6515 (2.67*10^-17)	0.6447 (7.3*10^-17)	0.6154 (5.40*10^-12)
Correlation (p-value) Permutation	0.5641 (1.86*10^-12)	0.6622 (5.3*10^-18)	0.6971 (1.62*10^-20)	0.7968(3.12*10^-30)	0.6425 (1*10^-16)	0.6726 (3.72*10^-13)

Tabella 2 Pearson Corelation e p-value Decision Tree

Anche in questo caso possimao valutare meglio i risultati disegnando dei grafici che utilizza i valori medi.

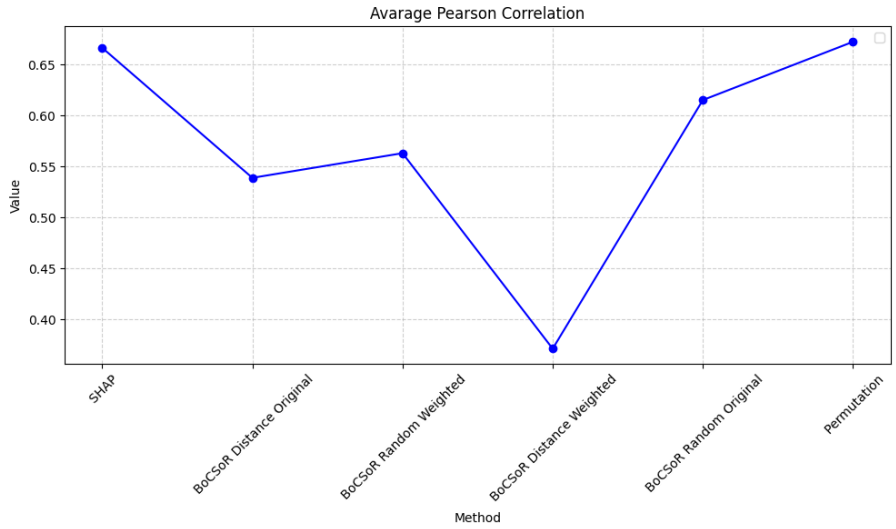

Figura 2 Grafico Pearson Correlation Decision Tree

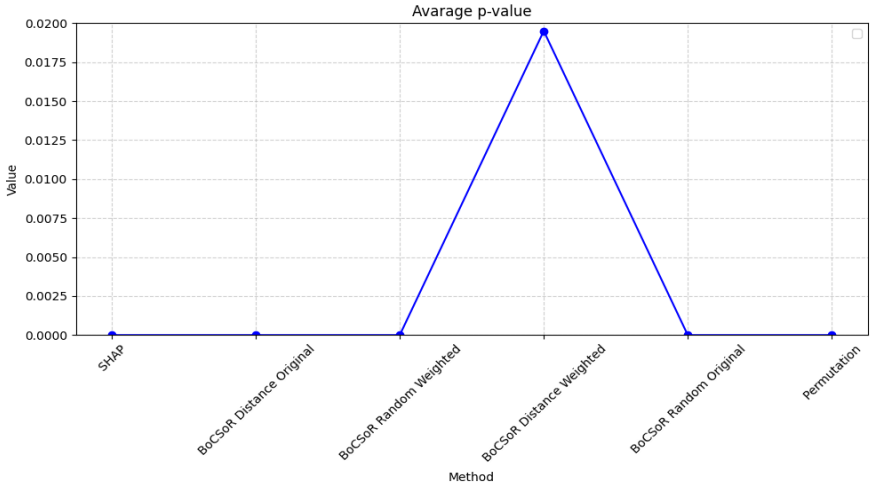

Figura 3 Grafico p-value Decision Tree

Dai grafici vediamo che, oltre a BoCSoR Distance Wegihted che ha una correlazione bassa legata ad un p-value più alto degli altri, ma comunque minore a 0.05, tutti i metodi restituiscono una buona correlazione di Pearson e p-value molto bassi, tutti inferiori all'ordine di 10^-6. Come ci aspettavamo SHAP e Permutation Importance hanno la correlazione maggiore con Intrinsic, ma i metodi BoCSoR, restituiscono una correlazione maggiore di 0.5, BoCSoR Random Original supera 0.6 per esempio. Quindi otteniamo buoni risultati con tempi di calcolo inferiori, provando a calcolare l'indice citato precedentemente ottengo:

IndiceSHAP = 0.618

IndiceBoCSoR_DO = 0.588

IndiceBoCSoR_RW = 0.607

IndiceBoCSoR_DW = 0.544

IndiceBoCSoR_RO = 0.621

IndicePermutation = 0.621

Anallizzando questi indici vediamo che i metodi di BoCSoR Random Weighted e BoCSoR Random Original hanno un'affidabilità comparabile con quella di SHAP e Permutation.

5.3 Random Forest

L'analisi che andiamo a fare usando il Random Forest è più precisa, possiamo infatti vedere come evolvono i risultati utilizzando un numero di alberi differente. Il dataset è stato processato in quattro modi diversi, le misure fatte con 5, 10 e 15 alberi non avranno una grande precisione ma permettono un calcolo in tempi accessibili, le misure fatte con 50 alberi restituiranno valori più precisi ma i tempi di calcolo cresceranno notevolmente.

5.3.1 5 Trees

Iniziamo valutando i risultati ottenuti con una foresta di 5 alberi. In questo caso i tempi sono ridotti e i risultati sono più approssimativi e di conseguenza anche la correlazione e il p-value saranno peggiori rispetto agli altri.

Random Forest (5 trees)						
Method	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Computational Time SHAP	318,6582	302,0701	304,0549	302,8725	307,9145	307,1140
Computational Time BoCSoR Distance Original	211,9699	174,1221	240,9312	200,3536	235,2526	212,5259
Computational Time BoCSoR Random Weighted	83,4326	78,4335	82,5870	83,3539	79,4270	81,4468
Computational Time BoCSoR Distance Weighted	134,5509	129,8727	133,5749	132,2960	125,0275	131,0644
Computational Time BoCSoR Random Original	80,3556	78,3402	84,6630	81,4365	81,7972	81,3185
Computational Time Permutation	194,3368	191,6900	207,0827	194,2470	186,2836	194,7280
Computational Time Intrinsic	3,9577*10^-5	2,5749*10^-5	2,7894*10^-5	3,4332*10^-5	4,1246*10^-5	3,33759*10^-5
Correlation (p-value) SHAP	0.5058 (6.13*10^-10)	0.6027 (2.08*10^-14)	0.5928 (6.96*10^-14)	0.4681 (1.51*10^-08)	0.5658 (1.55*10^-12)	0.5471 (3.14*10^-09)
Correlation (p-value) BoCSoR Distance Original	0.4032 (1.64*10^-06)	0.5262 (9.17*10^-11)	0.5376 (3.01*10^-11)	0.3404 (6.51*10^-05)	0.4692 (1.39*10^-08)	0.4553 (1.34*10^-05)
Correlation (p-value) BoCSoR Random Weighted	0.4853 (3.68*10^-09)	0.4469 (7.83*10^-08)	0.5179 (2.02*10^-10)	0.3897 (3.86*10^-06)	0.4723 (1.08*10^-08)	0.4624 (7.90*10^-07)
Correlation (p-value) BoCSoR Distance Weighted	0.3191 (1.92*10^-04)	0.4144 (7.82*10^-07)	0.3189 (1.94*10^-04)	0.3382 (7.31*10^-05)	0.4089 (1.13*10^-06)	0.3599 (9.21*10^-05)
Correlation (p-value) BoCSoR Random Original	0.4438 (9.80*10^-08)	0.5095 (4.39*10^-10)	0.5127 (3.29*10^-10)	0.2731 (1.53*10^-03)	0.3448 (5.15*10^-05)	0.4168 (3.17*10^-04)
Correlation (p-value) Permutation	0.5794 (3.38*10^-13)	0.6794 (3.40*10^-19)	0.5549 (5.05*10^-12)	0.5039 (7.31*10^-10)	0.5466 (1.21*10^-11)	0.5728 (1.50*10^-10)

Tabella 3 Tempi computazioneli, Pearson Correlation e p-value Random Forest (5 trees)

Come prima, disegnando grafici utilizzando i valori medi ci permette di osservare meglio le differenze tra i vari metodi.

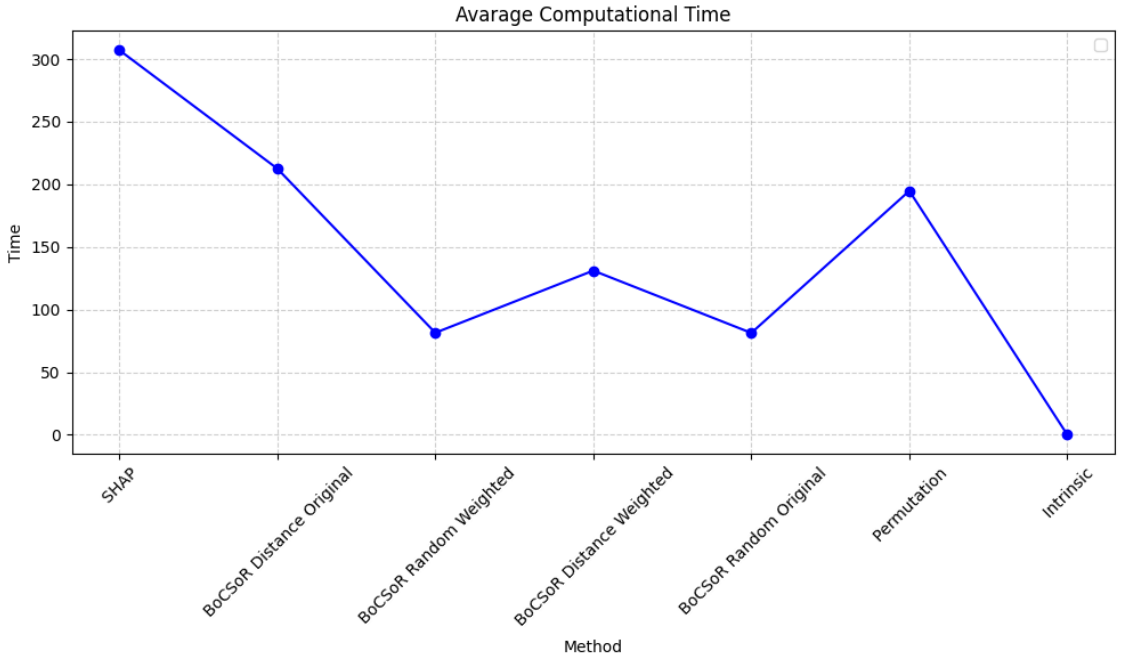

Figura 4 Grafico tempi computazionali Random Forest (5 trees)

Come ci aspettavamo i tempi sono maggiori rispetto ai tempi di calcolo misurati usando il Decision Tree. Il tempo necessario per calcolare le importance utilizzando il metodo BoCSoR Distance Original è aumentato particolarmente di più rispetto agli altri metodi.

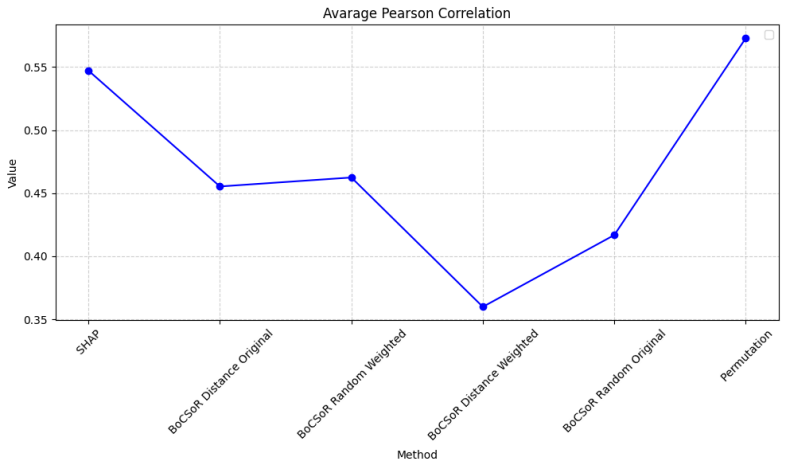

Figura 5 Grafico Pearson Correlation Random Forest (5 trees)

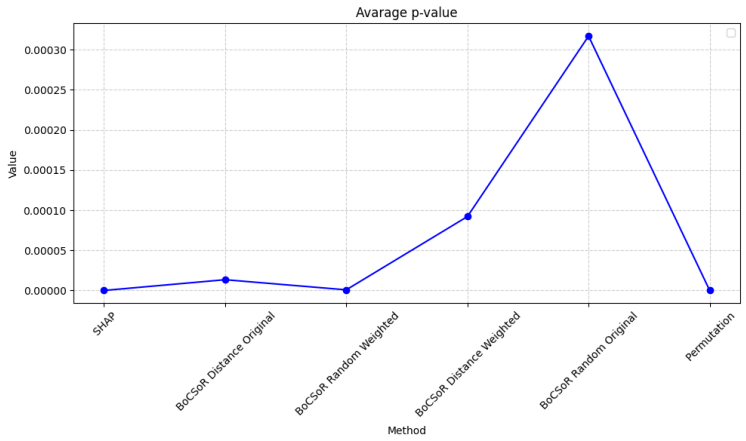

Figura 6 Grafico p-value Random Forest (5 trees)

In questo caso gli unici metodi che hanno una correlazione superiore a 0.5 sono SHAP e Permutation, nessun metodo BoCSoR ha quindi una correlazione che possiamo considerare buona. A livello di p-value otteniamo tutti p-value inferiori a 10^-3, quindi i risultati sono generalmente significativi. Ricalcolando gli indici otteniamo:

IndiceSHAP = 0.588

IndiceBoCSoR_DO = 0.565

IndiceBoCSoR_RW = 0.569

IndiceBoCSoR_DW = 0.542

IndiceBoCSoR_RO = 0.558

IndicePermutation = 0.595

Tutti gli indici calcolati sono minori rispetto i corrispondenti calcolati con il modello Decision Tree, confermando che l'utilizzo di soli 5 alberi rende una bassa affidabilità del metodo.

5.3.2 10 trees

Qui si vede un aumento dei tempi computazionali con una conseguente evoluzione delle correlazioni e dei p-value.

Random Forest (10 trees)						
Method	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Computational Time SHAP	327,2473	316,8250	326,4350	320,5749	330,5301	324,3224
Computational Time BoCSoR Distance Original	605,6581	602,5048	619,9881	574,4015	612,2937	602,9692
Computational Time BoCSoR Random Weighted	107,0754	110,9540	109,4721	107,2995	105,4517	108,0505
Computational Time BoCSoR Distance Weighted	160,0415	168,7298	164,4066	158,4890	164,9395	163,3213
Computational Time BoCSoR Random Original	114,8576	105,7342	104,0967	103,0767	104,0108	106,3552
Computational Time Permutation	413,3261	407,7089	415,8161	403,5278	441,4218	416,3601
Computational Time Intrinsic	2,5749*10^-5	3,4093*10^-5	3,6716*10^-5	3,4332*10^-5	3,9577*10^-5	3,4093*10^-5
Correlation (p-value) SHAP	0.4010 (1.89*10^-06)	0.5426 (1.80*10^-11)	0.3949 (2.78*10^-06)	0.3400 (6.65*10^-05)	0.4830 (4.48*10^-09)	0.4323 (1.42*10^-05)
Correlation (p-value) BoCSoR Distance Original	0.4963 (1.43*10^-09)	0.6271 (8.68*10^-16)	0.6242 (1.28*10^-15)	0.3906 (3.65*10^-06)	0.6182 (2.86*10^-15)	0.5513 (7.31*10^-07)
Correlation (p-value) BoCSoR Random Weighted	0.3233 (1.56*10^-04)	0.4678 (1.55*10^-08)	0.2716 (1.63*10^-03)	0.1171 (1.81*10^-01)	0.2635 (2.26*10^-03)	0.2887 (3.70*10^-02)
Correlation (p-value) BoCSoR Distance Weighted	0.1807 (3.82*10^-02)	0.4884 (2.84*10^-09)	0.3722 (1.11*10^-05)	0.4142 (7.95*10^-07)	0.3380 (7.39*10^-05)	0.3587 (7.65*10^-03)
Correlation (p-value) BoCSoR Random Original	0.3120 (2.70*10^-04)	0.3604 (2.19*10^-05)	0.2314 (7.59*10^-03)	0.1167 (1.83*10^-01)	0.2024 (1.99*10^-02)	0.2446 (4.21*10^-02)
Correlation (p-value) Permutation	0.4092 (1.10*10^-06)	0.5060 (6.05*10^-10)	0.4693 (1.38*10^-08)	0.4330 (2.15*10^-07)	0.4868 (3.25*10^-09)	0.4609 (2.67*10^-07)

Tabella 4 Tempi computazionali, Pearson Correlation e p-value Random Forest (10 trees)

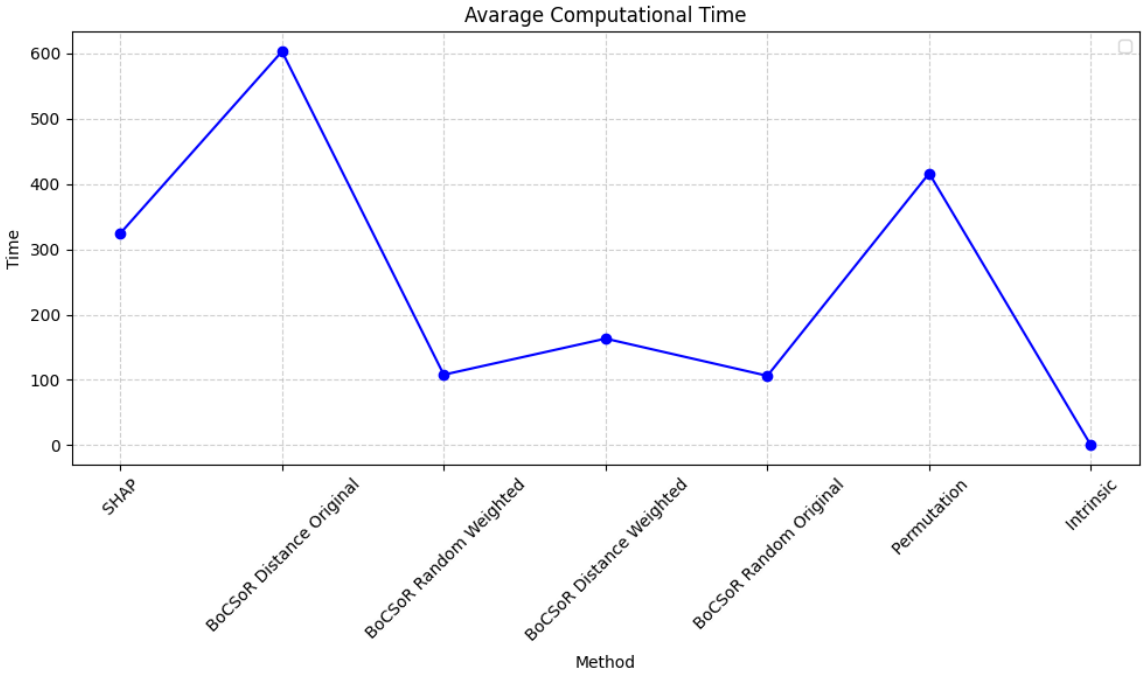

Figura 7 Grafico tempi computazionali Random Forest (10 trees)

Si osserva già, come prima, un notevole aumento del tempo necessario per BoCSoR Distance Original; infatti, il suo tempo di calcolo supera il tempo di cui ha bisogno SHAP per calcolare le importanze.

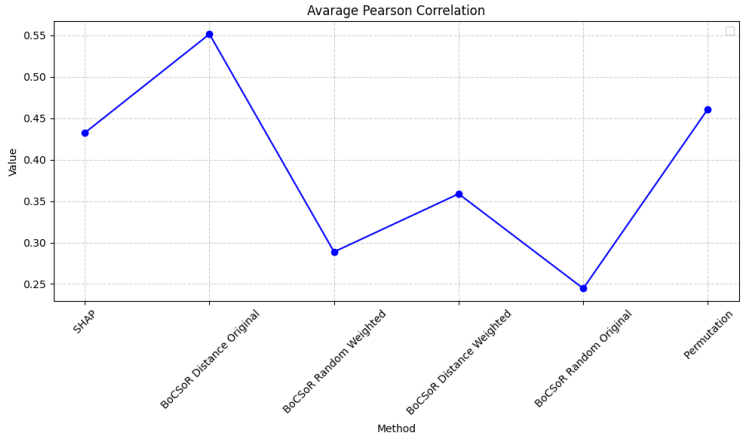

Figura 8 Grafico Pearson Correlation Random Forest (10 trees)

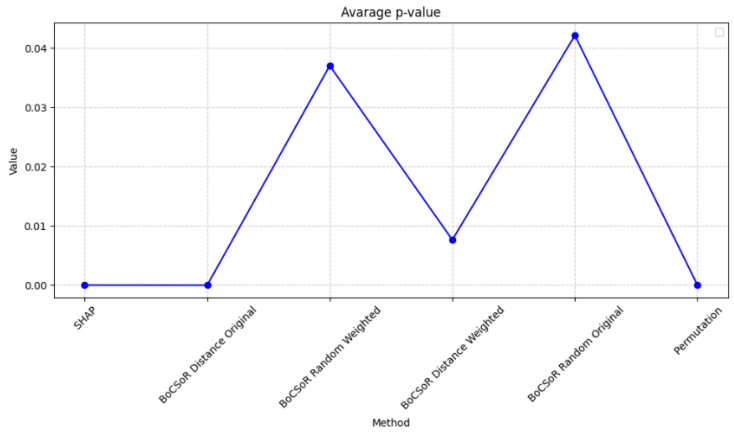

Figura 9 Grafico p-value Random Forest (10 trees)

Seguendo l'andamento dei tempi, anche la Pearson Correlation di BoCSoR Distance Originale supera la correlazione di SHAP e Permutation, con un p-value molto basso, al pari del p-value di Permutation Importance.

IndiceSHAP = 0.559

IndiceBoCSoR_DO = 0.588

IndiceBoCSoR_RW = 0.518

IndiceBoCSoR_DW = 0.540

IndiceBoCSoR_RO = 0.506

IndicePermutation = 0.566

Gli indici calcolati rimangono molto simili ai precedenti, con colo piccole modifiche.

5.3.3 15 trees

Random Forest (15 trees)						
Method	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Computational Time SHAP	318,7729	309,0864	312,1513	308,1432	310,6463	311,7600
Computational Time BoCSoR Distance Original	696,7935	707,2661	713,3542	700,6128	705,7840	704,7621
Computational Time BoCSoR Random Weighted	124,0111	120,9891	119,5327	122,6784	123,0462	122,0515
Computational Time BoCSoR Distance Weighted	167,7899	172,0061	166,4853	170,3168	167,1192	168,7435
Computational Time BoCSoR Random Original	123,3575	123,3753	120,8569	122,0647	120,1486	121,9606
Computational Time Permutation	437,8882	470,9687	463,5632	429,8362	431,7163	446,7945
Computational Time Intrinsic	3,5047*10^-5	2,2649*10^-5	2,4557*10^-5	2,3365*10^-5	4,7720*10^-5	3,0667*10^-5
Correlation (p-value) SHAP	0.4612 (2.61*10^-08)	0.5886 (1.15*10^-13)	0.4708 (1.22*10^-08)	0.5024 (8.34*10^-10)	0.5731 (6.90*10^-13)	0.5192 (7.84*10^-09)
Correlation (p-value) BoCSoR Distance Original	0.5805 (2.97*10^-13)	0.7079 (2.30*10^-21)	0.6452 (6.80*10^-17)	0.5231 (1.23*10^-10)	0.7265 (6.24*10^-23)	0.6367 (2.47*10^-11)
Correlation (p-value) BoCSoR Random Weighted	0.2543 (3.25*10^-03)	0.1913 (2.80*10^-02)	0.1080 (2.18*10^-01)	0.0982 (2.62*10^-01)	0.3599 (2.25*10^-05)	0.2024 (1.02*10^-01)
Correlation (p-value) BoCSoR Distance Weighted	0.3417 (6.08*10^-05)	0.2446 (4.70*10^-03)	0.3389 (7.05*10^-05)	0.1982 (2.27*10^-02)	0.4045 (1.50*10^-06)	0.3056 (5.51*10^-03)
Correlation (p-value) BoCSoR Random Original	0.2912 (7.07*10^-04)	0.2063 (1.77*10^-02)	0.2293 (8.19*10^-03)	0.2440 (4.82*10^-03)	0.2748 (1.43*10^-03)	0.2491 (6.56*10^-03)
Correlation (p-value) Permutation	0.4678 (1.55*10^-08)	0.5877 (1.28*10^-13)	0.5180 (2.01*10^-10)	0.4964 (1.42*10^-09)	0.6294 (6.30*10^-16)	0.5399 (3.43*10^-09)

Tabella 5 Tempi computazioni, Pearson Correlation e p-value Random Forest (15 trees)

Si nota un andamento delle tempistiche e delle correlazioni dei vari metodi.

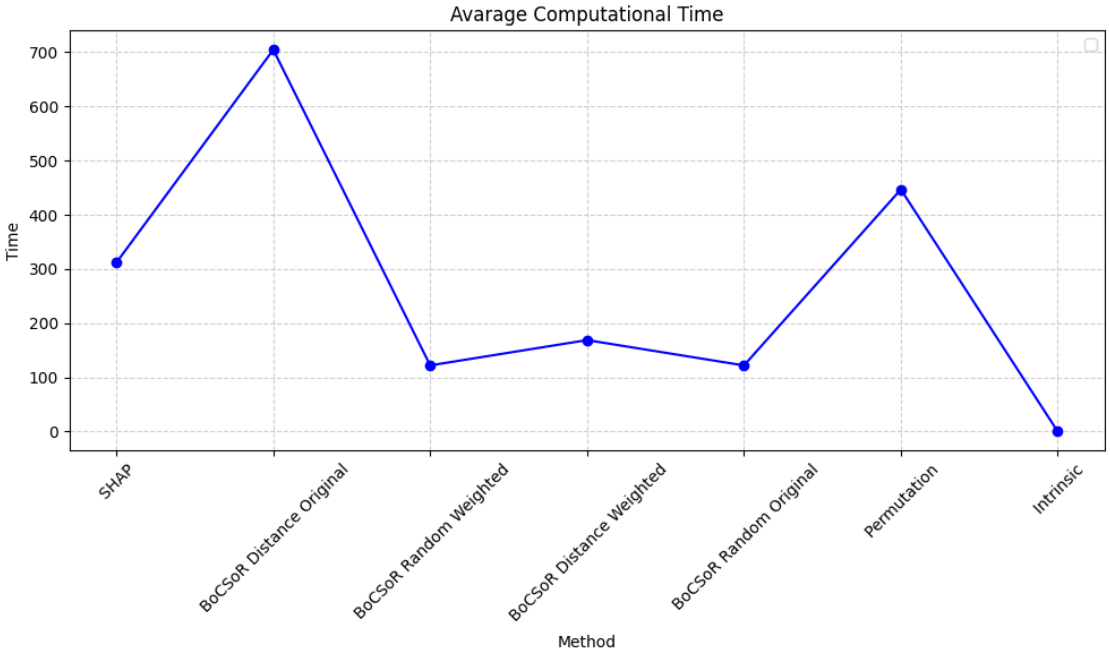

Figura 10 Grafico tempi computazionali Random Forest (15 trees)

Il grafico è molto simile al precedente, visto con l'utilizzo di 10 trees, si vede che BoCSoR Distance Original è il metodo che ha bisogno sempre di più tempo per calcolare le importanze, a differenza degli altri che hanno un aumento più lieve.

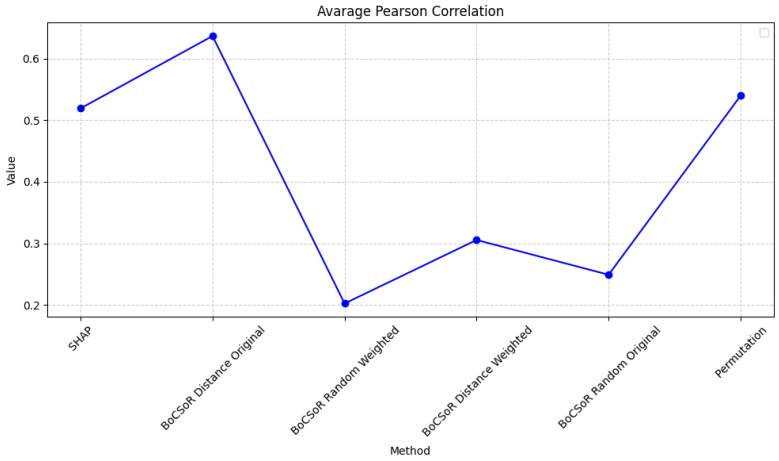

Figura 11 Grafico Pearson Correlation Random Forest (15 trees)

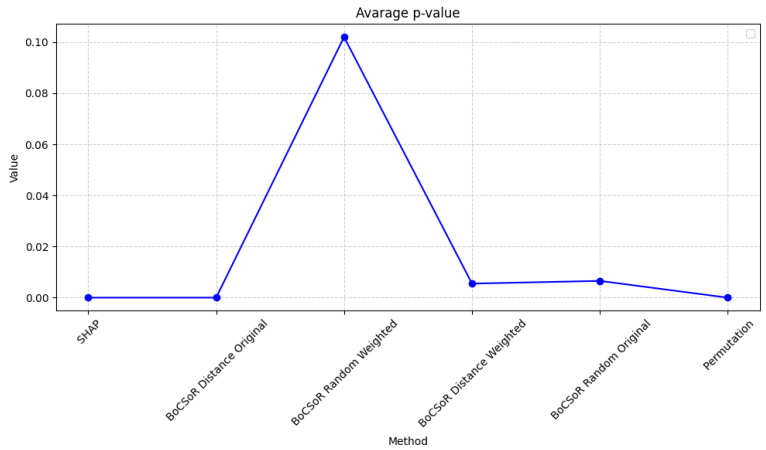

Figura 12 Grafico p-value Random Forest (15 trees)

Come prima vediamo un aumento della correlazione di Pearson per BoCSoR Distance Original che segue l'aumento del tempo computazionale.

Si nota un andamento che riguarda sia i tempi computazionali sia le correlazioni, SHAP e Permutation tendono a mantenere entrambi i parametri stabili, senza grandi modifiche. I metodi BoCSoR hanno una notevole diminuzione della correlazione con l'aumento degli alberi. Il metodo che fa eccezione è BoCSoR Distance Original, che ad un importante aumento dei tempi di calcolo segue un aumento particolare della correlazione.

IndiceSHAP = 0.581

IndiceBoCSoR_DO = 0.610

IndiceBoCSoR_RW = 0.485

IndiceBoCSoR_DW = 0.527

IndiceBoCSoR_RO = 0.513

IndicePermutation = 0.586

L'andamento sopracitato lo osserviamo anche calcolando gli indici; infatti SHAP e Permutation sono rimasti stabili, BoCSoR Distance Original va aumentando mentre gli altri metodi BoCSoR hanno indici diminuenti.

5.3.4 50 trees

Utilizzando 50 alberi andiamo ad aumentare molto la precisione del calcolo a discapito dei tempi richiesti che aumentano anch'essi notevolmente.

Random Forest (50 trees)						
Method	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Computational Time SHAP	416,8842	400,3290	418,2898	401,4638	407,1595	408,8252
Computational Time BoCSoR Distance Original	1720,3982	1689,5116	1781,4640	1691,3631	1701,6066	1.716,8687
Computational Time BoCSoR Random Weighted	305,2483	310,7851	298,1135	293,1542	298,4750	301,1552
Computational Time BoCSoR Distance Weighted	339,5475	358,4904	353,4282	342,9668	343,3540	347,5574
Computational Time BoCSoR Random Original	292,8579	306,8225	306,0105	298,4015	297,5801	300,3345
Computational Time Permutation	1099,3042	1314,1319	1296,4534	1189,4014	1246,6998	1.229,1981
Computational Time Intrinsic	4,2438*10^-5	2,7418*10^-5	3,6001*10^-5	2,6941*10^-5	2,6226*10^-5	3,1308*10^-5
Correlation (p-value) SHAP	0.6015 (2.41*10^-14)	0.6289 (6.74*10^-16)	0.5093 (4.50*10^-10)	0.5471 (1.15*10^-11)	0.6437 (8.48*10^-17)	0.5861 (9.22*10^-11)
Correlation (p-value) BoCSoR Distance Original	0.7064 (3.01*10^-21)	0.7638 (1.76*10^-26)	0.7387 (4.98*10^-24)	0.6614 (6.06*10^-18)	0.7593 (5.04*10^-26)	0.7259 (1.21*10^-18)
Correlation (p-value) BoCSoR Random Weighted	0.2291 (8.23*10^-03)	0.1953 (2.48*10^-02)	0.2188 (1.17*10^-02)	0.1509 (8.42*10^-02)	0.1785 (7.41*10^-02)	0.1945 (5.55*10^-02)
Correlation (p-value) BoCSoR Distance Weighted	0.0525 (5.50*10^-01)	0.3159 (2.25*10^-04)	0.1094 (2.12*10^-01)	0.2376 (6.09*10^-03)	0.2217 (1.06*10^-02)	0.1874 (1.56*10^-01)
Correlation (p-value) BoCSoR Random Original	0.2095 (1.59*10^-02)	0.1959 (2.44*10^-02)	0.1142 (1.92*10^-01)	0.0943 (2.82*10^-01)	0.1996 (2.17*10^-02)	0.1627 (1.07*10^-01)
Correlation (p-value) Permutation	0.4832 (4.40*10^-09)	0.5089 (4.64*10^-10)	0.4055 (1.42*10^-06)	0.5169 (2.21*10^-10)	0.5180 (2.01*10^-10)	0.4865 (2.84*10^-07)

Tabella 6 Tempi computazionali, Pearson Correlation e p-value Random Forest (50 trees)

Osserviamo i grafici generati per capire se l'andamento notato nei paragrafi precedenti si ripete nuovamente. Già dalla tabella si può comunque affermare che l'andamento è stato mantenuto.

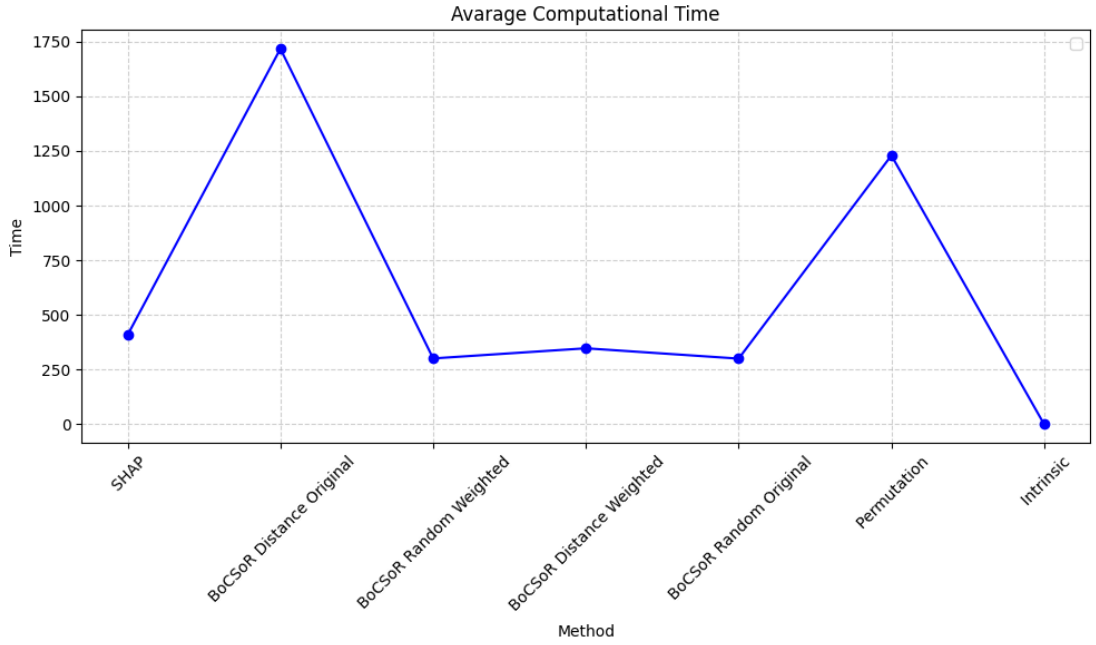

Figura 13 Grafico tempi computazionali Random Forest (50 trees)

Il grafico generato dai valori medi dei tempi computazionali è sempre simile hai precedenti. Notiamo, infatti, che il primo metodo BoCSoR è sempre quello con un aumento maggiore rispetto agli altri, in questo caso però è accompagnato da Permutation che richiede anch'esso notevoli risorse di tempo utilizzando un modello con 50 alberi.

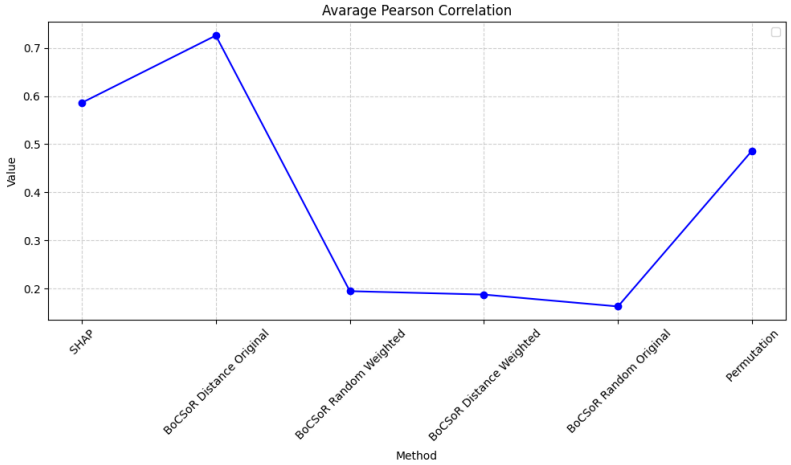

Figura 14 Grafico Pearson Correlation Random Forest (50 trees)

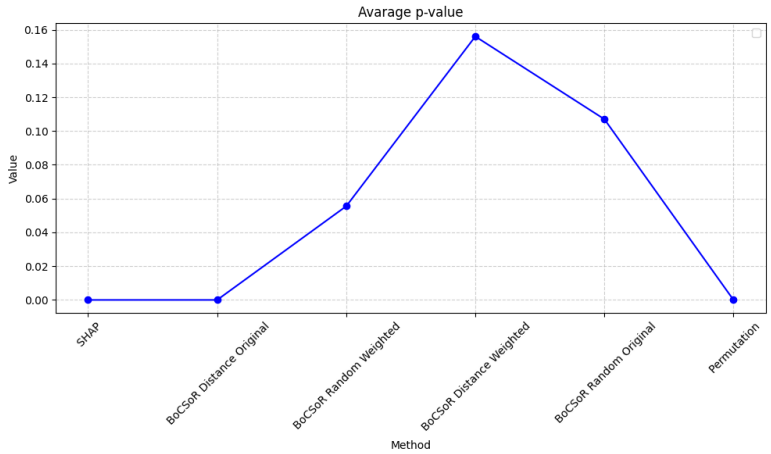

Figura 15 Grafico p-value Random forest (50 trees)

L'andamento sembra essere confermato anche aumentando il numero di alberi; infatti, anche in questo caso la correlazione maggiore si ottiene con il primo metodo BoCSoR, legato anche ad un p-value estremamente piccolo, nell'ordine di 10^-18, ad indicare una grande significatività dei risultati ottenuti. Allo stesso tempo gli altri metodi BoCSoR ottengono delle correlazioni sempre minori insieme ad un aumento del valore del p-value.

IndiceSHAP = 0.597

IndiceBoCSoR_DO = 0.632

IndiceBoCSoR_RW = 0.489

IndiceBoCSoR_DW = 0.471

IndiceBoCSoR_RO = 0.472

IndicePermutation = 0.572

Gli indici risultanti da quest'ultimo calcolo mostrano perfettamente che aumentando il numero di alberi, l'andamento osservato precedentemente permane. BoCSoR Distance Original ottiene l'indice maggiore visto in tutte le misurazioni, considerando anche il modello Decision Tree, mentre gli altri metodi BoCSoR hanno degli indici che vanno

diminuendo con l'aumento del numero degli alberi. SHAP e Permutation mantengono la caratteristica di riuscire a mantenere un indice stabile.

6. Conclusioni

Dopo aver calcolato le importanze ripetute volte con metodi diversi e con modelli diversi, ora possiamo concludere se utilizzare il nuovo metodo ibrido BoCSoR conviene rispetto all'utilizzo di metodi standard.

6.1 Analisi Parametri

Analizziamo ora i risultati considerando i singoli parametri, per avere una valutazione precisa su ogni caratteristica.

6.1.1 Tempo computazionale

Considerando il tempo di calcolo necessario per calcolare le importanze si può affermare che, in generale, BoCSoR è più conveniente rispetto a SHAP e Permutation Importance.

Guardando i risultati ricevuti utilizzando il Decision Tree osserviamo che BoCSoR e nettamente meglio, impiega in tutti i casi meno tempo rispetto ai metodi standardizzati, rendendolo quindi molto utile per processare dataset molto ampi.

Guardando, invece, i risultati ottenuti utilizzando la Random Forest la questione cambia, infatti il metodo BoCSoR Distance Original, quando andiamo ad utilizzare un numero di alberi maggiore o uguale a 10, supera in tempo di calcolo sia SHAP che Permutation Importance, risultando quindi sconveniente in caso di risorse temporali ridotte.

6.1.2 Pearson Correlation

Prendendo in analisi la Pearson Correlation che ogni metodo ha con Intrinsic, come ptevamo aspettarci, generalmente sono SHAP e Permutation Importance ad avere valori migliori rispetto a BoCSoR.

Osservando i risultati ottenuti utilizzando il Decision Tree vediamo che SHAP e Permutation hanno valori migliori, ma non con un distacco netto da quelli ottenuti utilizzando BoCSor.

Osservando i risultati ottenuti con la Random Forest, invece, abbiamo anche qui un'inversione rispetto al Decision Tree quando utilizziamo un numero maggiore o uguale a 10 alberi. Da quel momento in poi la correlazione di BoCSoR Distance Original diventa più alta rispetto alle correlazioni misurate con SHAP e Permutation, con un distacco che va aumentando con l'aumentare del numero di alberi utilizzati.

6.1.3 Indice unificante

C'è da dire inizialmente che questo indice utilizza dei pesi arbitari, secondo l'importanza, dei parametri precedentemente valutati, considerata idonea per questo studio, come spiegato nel paragrafo 5.1.3. Variando il peso che viene dato ad ogni singolo parametro si ottengono valori diversi.

Osservando i valori calcolati dei vari metodi otteniamo un riassunto di ciò che abbiamo commentato riguardo il computational time e la Pearson Correlation.

Gli indici che riguardano il modello Decision Tree mostrano che, indicativamente, i migliori metodi da utilizzare sono Permutation Importance e BoCSoR Random Original.

Gli indici riguardanti il modello mostrano un andamento ben preciso. Con l'aumentare del numero di alberi considerati SHAP e Permutation Importance restano stabili, mantengono circa lo stesso valore; invece, i metodi BoCSoR hanno un indice che va diminuendo, quindi si può dire che hanno un comportamento inversamente proporzionale rispetto al numero di alberi. L'unico metodo che va contro a quest'ultimo andamento è BoCSoR Distance Original che aumenta all'aumentare del numero di alberi, il che lo rende la scelta migliore per foreste sempre più complesse.

6.2 Conclusioni finali

Dopo aver analizzato i risultati emersi dallo studio, sorgono domande cruciali riguardo l'affidabilità dei nuovi metodi di XAI, come BoCSoR, in confronto a metodi ormai consolidati come SHAP e Permutation Importance. Questi ultimi rappresentano standard di riferimento nel campo della spiegabilità dei modelli di intelligenza artificiale, poiché offrono spiegazioni umanamente comprensibili che colmano il divario tra la complessità algoritmica e le esigenze di interpretazione degli esperti di dominio. Tuttavia, la loro applicazione presenta sfide significative, soprattutto in termini di risorse computazionali necessarie, che ne limitano l'utilizzo in scenari con grandi volumi di dati o vincoli di tempo.

In questo contesto, il metodo BoCSoR si propone come un'alternativa innovativa, con l'obiettivo di combinare affidabilità ed efficienza. L'idea centrale è sviluppare un approccio che non solo sia in grado di fornire spiegazioni attendibili, ma che lo faccia con un costo computazionale significativamente inferiore rispetto ai metodi tradizionali. Idealmente, BoCSoR potrebbe persino superare questi metodi in affidabilità, mantenendo al contempo un costo computazionale accettabile.

Dai risultati ottenuti possiamo osservare che l'utilizzo di BoCSoR si dimostra più efficiente rispetto a SHAP e Permutation Importance in due scenari distinti: con modelli decisionali semplici, come i Decision Tree, e con modelli complessi, come le Random Forest. Nel caso dei Decision Tree, BoCSoR offre un buon livello di affidabilità, leggermente inferiore rispetto ai metodi standard, ma con un significativo vantaggio in termini di tempi di esecuzione. Questo lo rende particolarmente adatto per applicazioni che richiedono decisioni rapide senza compromettere eccessivamente la qualità delle spiegazioni.

Per quanto riguarda le Random Forest, la performance di BoCSoR dipende dal numero di alberi utilizzati nel modello. Con un numero ridotto di alberi, l'efficacia dei metodi è comparabile; tuttavia, man mano che il numero di alberi cresce, BoCSoR emerge come il metodo più efficiente. In questi casi, non solo offre spiegazioni in tempi accettabili, ma lo fa mantenendo un livello di affidabilità paragonabile, se non superiore, ai metodi standard. Questo comportamento lo rende ideale per scenari in cui la complessità del modello aumenta, come nel caso di grandi dataset o sistemi con molteplici variabili.

In conclusione, l'introduzione di BoCSoR nel panorama dei metodi di XAI è pienamente giustificata. Questo approccio rappresenta un significativo passo avanti, permettendo agli utenti di ottenere spiegazioni affidabili in tempi ridotti. Inoltre, offre la flessibilità di adattarsi a diversi scenari: può essere utilizzato per ottenere risultati rapidi e accettabili, oppure per raggiungere livelli di affidabilità più elevati senza incorrere nei costi computazionali proibitivi tipici di metodi come SHAP. Di conseguenza, BoCSoR si dimostra una soluzione promettente per affrontare le sfide della spiegabilità nei modelli di

intelligenza artificiale, garantendo un equilibrio ottimale tra efficienza e qualità delle spiegazioni.

7. Appendice A: codice utilizzato

```
import pandas as pd
import numpy as np
from scipy.stats import pearsonr
import math
import itertools
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import StratifiedKFold
from catboost import CatBoostClassifier
import shap
from sklearn.tree import DecisionTreeClassifier, ExtraTreeClassifier
from BoCSoR import BoCSoR
from CounterfactualExplainerByProximity import CounterfactualExplainerByProximity
from sklearn.inspection import permutation_importance
import time
from DataLoader import DataLoader
from openpyxl import load_workbook
from openpyxl.styles import PatternFill
import lightgbm as lgb
import warnings


warnings.simplefilter("ignore")



class FeatureImportanceClassification:
    def __init__(self, datasets, repetitions=5,
output_file="feature_importance_results_classification.xlsx",
                 num_samples=100):
        self.datasets = datasets
        self.repetitions = repetitions
```

```
        self.output_file = output_file


    def model_setup(self, model_name, feature_names, categorical_features, X_train,
y_train):
        model = []
        intrinsic_feature_importance = {feature: 0.0 for feature in feature_names}
        if model_name == "CatBoost":
            params = {
                'border_count': 16,
                'depth': 10,
                'iterations': 100,
                'l2_leaf_reg': 5,
                'learning_rate': 0.01,
                'silent': True,
                'cat_features': categorical_features,
                'random_seed': 4
            }
            model = CatBoostClassifier(**params).fit(X_train, y_train)
            intrinsic_importance = model.get_feature_importance(type="FeatureImportance")


        elif model_name == "DecisionTree":
            params = {
                'criterion': "gini",  # "entropy", "log_loss"
                'min_samples_split': 4,
                'min_samples_leaf': 2,
                'max_leaf_nodes': 128,
                'random_state': 7
            }
            model = DecisionTreeClassifier(**params).fit(X_train, y_train)
            intrinsic_importance = model.feature_importances_


        elif model_name == "RandomForest":
```

```
            model = RandomForestClassifier(random_state=3).fit(X_train, y_train)
            intrinsic_importance = model.feature_importances_

        elif model_name == "ExtraTree":
            model = ExtraTreeClassifier(random_state=3).fit(X_train, y_train)
            intrinsic_importance = model.feature_importances_

        elif model_name == "LightGBM":
            params = {
                'learning_rate': 0.5,
                'max_depth': 2,
                'min_data_in_leaf': 5,
                'n_estimators': 50,
                'num_leaves': 10,
                'extra_trees': True,
                'verbosity': -1,
                'random_state': 3
            }
            model = lgb.LGBMClassifier(**params).fit(X_train, y_train)
            intrinsic_importance = model.feature_importances_

        elif model_name == "SVC":
            model = svm.SVC(kernel='linear', probability=True, random_state=3).fit(X_train,
y_train)
            intrinsic_importance = model.coef_

        else:
            print("Nome del modello non valido!")

        return model, intrinsic_importance

    def calculate_importances(self, model_name):
```

```
    # DataFrames per memorizzare i risultati
    feature_importance_df = pd.DataFrame()
    time_df = pd.DataFrame()

    dataset_to_features = {}

    for dataset_name in self.datasets:
        features, labels, numerical_features, categorical_features, column_names =
DataLoader.uci_dataset_handler(
            dataset_name)
        classes = np.unique(labels)

        kf = StratifiedKFold(n_splits=self.repetitions)

        feature_names = column_names[:-1]  # TODO: l'ultima è sempre la label nel
DataLoader
        dataset_to_features[dataset_name] = feature_names

        all_importance_data = []
        all_time_data = []

        for fold_num, (train_index, test_index) in enumerate(kf.split(features, labels),
start=1):
            X_train, X_test = features.iloc[train_index], features.iloc[test_index]
            y_train, y_test = labels[train_index], labels[test_index]

            [model, intrinsic_importance] = self.model_setup(model_name, feature_names,
categorical_features,
                                                             X_train, y_train)

            # Dizionari per memorizzare importanza delle caratteristiche e tempi
            feature_importances = {}
            times = {}
```

```
            # Importanza Intrinseca delle Caratteristiche
            start_time = time.time()
            feature_importances['Intrinsic'] = dict(zip(feature_names,
intrinsic_importance))
            times['Intrinsic'] = time.time() - start_time

            # SHAP con KernelExplainer
            start_time = time.time()
            shap_explainer = shap.KernelExplainer(model.predict_proba,
shap.sample(X_train, 100))
            shap_values = shap_explainer.shap_values(shap.sample(X_train, 100))
            shap_importance = np.abs(shap_values).mean(axis=0).mean(axis=1)
            feature_importances['SHAP'] = dict(zip(feature_names, shap_importance))
            times['SHAP'] = time.time() - start_time

            # Implementazioni di BoCSoR e altre tecniche...
            # (Puoi copiare e incollare il resto del codice nello stesso stile.)

        # Salva i risultati in Excel
        self.save_to_excel(feature_importance_df, time_df, dataset_to_features)

    def save_to_excel(self, feature_importance_df, time_df, dataset_to_features):
        with pd.ExcelWriter(self.output_file, engine='openpyxl') as writer:
            feature_importance_df.to_excel(writer, sheet_name='Feature Importance')
            time_df.to_excel(writer, sheet_name='Computation Times')

            # Applica colore
            workbook = writer.book
            feature_sheet = workbook['Feature Importance']
            time_sheet = workbook['Computation Times']
```

```
        # Genera colori per ogni dataset
        dataset_colors = self.generate_colors(len(self.datasets))

        # Applica colori al foglio Feature Importance
        # (Puoi continuare la traduzione qui per le sezioni successive.)

    def generate_colors(self, n):
        colors = []
        for i in range(n):
            color = f'{np.random.randint(0, 255):02X}{np.random.randint(0, 255):02X}{np.random.randint(0, 255):02X}'
            colors.append(color)
        return colors
def calcola_qualita(tempo, pearson_corr, p_value, peso_tempo=0.3, peso_corr=0.5, peso_pvalue=0.2):
    """
    Calcola un punteggio di qualità per un metodo basato su:
    - Tempo computazionale (minore è meglio)
    - Correlazione di Pearson (maggiore è meglio)
    - P-value (minore è meglio)

    :param tempo: Tempo computazionale (in secondi)
    :param pearson_corr: Correlazione di Pearson (valore tra -1 e 1)
    :param p_value: P-value (significatività statistica, minore è meglio)
    :param peso_tempo: Peso assegnato al tempo computazionale
    :param peso_corr: Peso assegnato alla correlazione di Pearson
    :param peso_pvalue: Peso assegnato al P-value
    :return: Punteggio di qualità (valore tra 0 e 1, maggiore è meglio)
    """
    # Normalizza i valori
    tempo_norm = 1 / (1 + tempo)  # Normalizza il tempo: valori più bassi ottengono punteggi più alti
```

```
    corr_norm = (pearson_corr + 1) / 2  # Normalizza la correlazione (-1,1) -> (0,1)
    pvalue_norm = 1 / (1 + p_value)  # Normalizza il p-value: valori più bassi ottengono punteggi più alti

    # Calcola il punteggio ponderato
    qualita = (peso_tempo * tempo_norm) + (peso_corr * corr_norm) + (peso_pvalue * pvalue_norm)
    return qualita

def main():
    # List of classification datasets and ML model available
    current_path = "BOCSOR 3.0/"
    datasets = ["BCI_2"]
    model_name = ["DecisionTree", "RandomForest"]

    # Number of repetitions (folds) for StratifiedKFold cross-validation
    repetitions = 5

    for m in model_name:
        for d in datasets:
            # Output Excel file to save the feature importance results
            output_file = current_path + "results/" + m + " " + d + "/feature_importance_results_classification_" + m + ".xlsx"

            # Instantiate the FeatureImportanceClassification class
            fic = FeatureImportanceClassification(datasets=datasets, repetitions=repetitions, output_file=output_file)

            # Calculate feature importances for a specific model
            fic.calculate_importances(m)

    # Carica il file Excel (specifica il percorso del file)
    file_path = output_file
```

```
try:
    df = pd.read_excel(file_path, sheet_name='Feature Importance')
except Exception as e:
    print(f"Errore nel caricamento del file Excel: {e}")
    df = None

if df is not None:
    # Indici delle colonne da escludere (aggiusta per l'indice base 0 di Python)
    colonne_da_escludere = [0, 8, 16, 24, 32, 40]

    # Seleziona le colonne da includere
    colonne_da_includere = [col for i, col in enumerate(df.columns) if i not in
colonne_da_escludere]

    # Nomi delle variabili da associare alle colonne
    nomi_variabili = [
        "Fold 1 Shap", "Fold 1 Intrinsic", "Fold 1 BoCSoR Distance Original",
        "Fold 1 BoCSoR Random Weighted", "Fold 1 BoCSoR Distance Weighted",
        "Fold 1 BoCSoR Random Original", "Fold 1 Permutation",
        "Fold 2 Shap", "Fold 2 Intrinsic", "Fold 2 BoCSoR Distance Original",
        "Fold 2 BoCSoR Random Weighted", "Fold 2 BoCSoR Distance Weighted",
        "Fold 2 BoCSoR Random Original", "Fold 2 Permutation",
        "Fold 3 Shap", "Fold 3 Intrinsic", "Fold 3 BoCSoR Distance Original",
        "Fold 3 BoCSoR Random Weighted", "Fold 3 BoCSoR Distance Weighted",
        "Fold 3 BoCSoR Random Original", "Fold 3 Permutation",
        "Fold 4 Shap", "Fold 4 Intrinsic", "Fold 4 BoCSoR Distance Original",
        "Fold 4 BoCSoR Random Weighted", "Fold 4 BoCSoR Distance Weighted",
        "Fold 4 BoCSoR Random Original", "Fold 4 Permutation",
        "Fold 5 Shap", "Fold 5 Intrinsic", "Fold 5 BoCSoR Distance Original",
        "Fold 5 BoCSoR Random Weighted", "Fold 5 BoCSoR Distance Weighted",
        "Fold 5 BoCSoR Random Original", "Fold 5 Permutation"
```

```
]

# Crea un dizionario per le variabili e associa ciascun array
variabili_dict = {}
for nome_var, colonna in zip(nomi_variabili, colonne_da_includere):
    variabili_dict[nome_var] = df[colonna].to_numpy()

# Variabili da confrontare
variabili_da_confrontare = [
    "Shap", "BoCSoR Distance Original", "BoCSoR Random Weighted",
    "BoCSoR Distance Weighted", "BoCSoR Random Original", "Permutation"
]

# Contenitore per salvare i risultati in formato tabella
risultati_formattati = {var: [] for var in variabili_da_confrontare}

# Ciclo sui fold
folds = [1, 2, 3, 4, 5]
medie_pearson = {}
medie_p_value = {}
for fold in folds:
    fold_intrinsic = variabili_dict.get(f"Fold {fold} Intrinsic")

    if fold_intrinsic is not None:
        for var in variabili_da_confrontare:
            array_da_confrontare = variabili_dict.get(f"Fold {fold} {var}")

            if array_da_confrontare is not None:
                # Calcola coefficiente di Pearson e p-value
                pearson_coef, p_value = pearsonr(fold_intrinsic, array_da_confrontare)
```

```
                # Formatta il risultato come "Pearson Coef (P-value)"
                risultato_formattato = f"{pearson_coef:.4f} ({p_value:.2e})"

                # Salva il risultato nella tabella
                risultati_formattati[var].append(risultato_formattato)

                # Accumula per le medie
                medie_pearson.setdefault(var, []).append(pearson_coef)
                medie_p_value.setdefault(var, []).append(p_value)

    # Calcola le medie e formatta
    medie_formattate = {}
    for var in variabili_da_confrontare:
        pearson_media = np.mean(medie_pearson[var]) if var in medie_pearson else
float('nan')
        p_value_media = np.mean(medie_p_value[var]) if var in medie_p_value else
float('nan')
        # Seleziona la riga dei tempi
        if riga_tempi < len(data):
            tempi = data.iloc[riga_tempi]  # Ottieni i valori della riga specificata
            tempi_media = tempi.mean()    # Calcola la media dei valori numerici

        punteggio = calcola_qualita(tempi_media, pearson_media, p_value_media)
        print(f"Punteggio di qualità: {punteggio:.3f}")

        medie_formattate[var] = f"{pearson_media:.4f} ({p_value_media:.2e})"

    # Aggiungi le medie come una nuova riga
    for var, media in medie_formattate.items():
        risultati_formattati[var].append(media)

    # Crea il DataFrame finale
```

```
colonne_finali = [f"Fold {fold}" for fold in folds] + ["Media"]
risultati_df = pd.DataFrame(risultati_formattati, index=colonne_finali).transpose()

# Scrivi i risultati su un file Excel
output_file = "risultati_pearson_formattati_con_medie.xlsx"
risultati_df.to_excel(output_file, sheet_name="Risultati Formattati")

print(f"Risultati salvati in '{output_file}'.")
```

8. Bibliografia

Matching the expert's knowledge via a counterfactual-based feature importance measure
- Alfeo A.L., Cimino M.G.C.A., Gagliardi G. - 2023

Peeking inside the black-box: a survey on explainable artificial intelligence.
- Adadi A., Berrada M. – 2018

A unified approach to interpreting model predictions.
- Lundberg S.M., Lee S.I. – 2017

Towards Rigorous Interpretations: a Formalisation of Feature Attribution
- Afchar D., Guigue V., Hennequin R. – 2021

Explainableartificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence. Information Fusion p. 101805
- Ali S., Abuhmed T., El-Sappagh S., Muhammad K., Alonso-Moral J.M., Confalonieri R., Guidotti R., Del Ser J., Diaz-Rodriguez N., Herrera F. - 2023

Permutation importance: a corrected feature importance measure. Bioinformatics 26(10), 1340–1347
- Altmann, A. Tolosi L., Sander O., Lengauer T. - 2010

Wikipedia – Pearson Correlation Coefficent

Wikipedia – P-value

stackoverflow.com – Calculating pearson correlation and significance in python