POLITECNICO DI TORINO

Corso di Laurea
Ingegneria Matematica

Tesi di Laurea

Study of Shapley Value-based Explainability in Machine Learning

Supervisori
Prof. Francesco Vaccarino
Dott. Antonio Mastropietro
firma dei supervisori

. . . . . .

. . . . . .

Candidato
Emanuele Pinna

firma del candidato

. . . . . .

Anno Accademico 2022-2023

Ai miei genitori
A mia zia Maria†

Abstract

La tesi "Study of Shapley Value-based Explainability in Machine Learning" affronta la cruciale questione dell'interpretabilità dei modelli di machine learning nell'ambito dell'intelligenza artificiale (IA). In un'epoca in cui l'IA svolge un ruolo fondamentale in svariati settori, la sua complessità e l'opacità dei modelli di machine learning hanno sollevato significative preoccupazioni riguardo alla trasparenza e alla comprensibilità di tali sistemi.
La tesi è studiata per affrontare una serie di obiettivi chiave: innanzitutto, si presenta un quadro generale dei principali sviluppi nell'ambito dell'IA e della sua crescente importanza nella società contemporanea, in secondo luogo, si sottolinea l'importanza di comprendere a fondo il funzionamento dei modelli di machine learning al fine di sfruttarne appieno i vantaggi e di mitigarne le sfide.
La tesi fornisce inoltre le basi matematiche e concettuali necessarie per una comprensione approfondita dei metodi di spiegazione, con un focus particolare sulla teoria dei giochi e sui concetti matematici fondamentali. Queste conoscenze costituiscono la base per la comprensione dei metodi SHAP (SHapley Additive exPlanations), che rappresentano il fulcro della ricerca.
Un'analisi esaustiva dei metodi SHAP è condotta per esaminarne la storia, il funzionamento e l'innovazione che portano nel campo dell'interpretabilità dei modelli di machine learning. L'obiettivo è comprendere le ragioni che rendono questi metodi così innovativi e discutere i possibili ambiti di applicazione.
Successivamente, la tesi passa alla fase sperimentale, in cui vengono condotti esperimenti mirati per valutare l'efficacia dei metodi SHAP confrontandoli con altri metodi di spiegazione, con un focus particolare sul metodo LIME (Local Interpretable Model-agnostic Explanations).
L'analisi dei risultati è approfondita al fine di valutare l'efficacia dei metodi SHAP in vari contesti, inclusi problemi di classificazione e regressione. Gli esperimenti includono dataset con e senza perturbazioni, nonché dataset con un numero variabile di feature. Inoltre, sono stati condotti esperimenti su dataset in cui le feature possono essere più o meno correlate, consentendo di esaminare come le prestazioni degli algoritmi variassero in queste specifiche condizioni.
In conclusione, la tesi "Study of Shapley Value based Explainability in Machine Learning" mira a esaminare dettagliatamente i metodi di spiegazione, concentrandosi sui metodi SHAP, e a valutare la loro applicabilità in contesti di shallow learning. Attraverso analisi rigorose e sperimentazioni, l'obiettivo è contribuire alla comprensione di come questi metodi possano migliorare la trasparenza e l'interpretabilità dei modelli di machine learning, promuovendo la conoscenza e l'innovazione in questo campo cruciale.

Ringraziamenti

Desidero dedicare questo piccolo spazio della mia tesi per esprimere la mia profonda gratitudine a tutte le persone che hanno contribuito in modo significativo alla realizzazione di questo percorso accademico e alla stesura di questa tesi. Senza il loro supporto, il mio viaggio attraverso l'università e la ricerca sarebbe stato molto più arduo.
Innanzitutto desidero ringraziare i miei genitori, che fin dall'inizio hanno sostenuto la mia decisione di frequentare il Politecnico di Torino dandomi la possibilità di vivere questa straordinaria esperienza. Il loro costante incoraggiamento, il loro affetto e il loro sostegno hanno reso il percorso accademico meno gravoso, permettendomi di concentrarmi sugli studi e sulle mie passioni.
Ringrazio tutta la mia famiglia, nonni, cugini, zii e parenti, che mi sono stati vicini in ogni fase di questo percorso. Le loro parole di incoraggiamento e il loro sostegno morale sono stati una fonte di forza in ogni momento, specialmente quando, al mio ritorno in Sardegna e in Sicilia, mi facevano sentire come se non me ne fossi mai realmente andato.
Non posso dimenticare di ringraziare i miei amici, sia quelli dell'università che quelli lontani. I miei compagni di corso sono stati una parte fondamentale del mio percorso accademico, condividendo gioie e sfide in aula e fuori. Gli amici lontani, nonostante la distanza, sono stati sempre presenti con il loro affetto e il loro calore.
Un ringraziamento ai miei coinquilini, che hanno reso questi due anni di vita condivisa un'esperienza indimenticabile. Le nostre risate, le cene insieme e le lunghe chiacchierate hanno reso la vita quotidiana più divertente e meno monotona.
Desidero inoltre esprimere la mia gratitudine al mio professore, Francesco Vaccarino, e al correlatore Antonio Mastropietro, che hanno guidato la mia ricerca e mi hanno fornito strumenti preziosi per ampliare il mio bagaglio culturale. La loro competenza e la loro disponibilità nel discutere temi di matematica e intelligenza artificiale hanno ispirato il mio interesse per questi campi, interesse che continua a crescere ogni giorno di più.
Infine, un ringraziamento speciale va a mio fratello, che è sempre stato una presenza costante nella mia vita. La sua spalla su cui poggiarmi e la sua disponibilità nel sostenermi durante i momenti più difficili sono stati inestimabili.
A tutti voi va il mio più sincero ringraziamento. Senza il vostro contributo e il vostro affetto, questo traguardo non sarebbe stato possibile. La vostra presenza e il vostro sostegno sono stati i pilastri su cui ho costruito il mio cammino accademico, e per questo vi sarò eternamente grato.

Indice

Elenco delle figure

If you wish success in life, make perseverance your bosom friend, experience your wise counselor, caution your elder brother and hope your guardian genius.
[J. Addison]

Capitolo 1

Introduzione

1.1 Artificial Intelligence

L'intelligenza artificiale (IA) è un campo della scienza e dell'informatica che mira a creare macchine in grado di svolgere attività che richiedono intelligenza umana. Negli ultimi decenni, l'IA ha raggiunto una notevole evoluzione, trasformando in modo significativo molti aspetti della nostra società e influenzando settori come la sanità, l'automazione industriale, l'e-commerce, i trasporti e molti altri. Il suo obiettivo fondamentale è quello di sviluppare sistemi che possano imitare o superare le capacità umane in termini di apprendimento, ragionamento, comprensione del linguaggio naturale, percezione sensoriale e decisioni intelligenti. Ciò viene realizzato attraverso l'utilizzo di algoritmi e modelli matematici che consentono alle macchine di apprendere dai dati, adattarsi all'ambiente circostante e migliorare le proprie prestazioni nel tempo.
Una delle aree chiave dell'IA è il *Machine Learning* (chiamato anche apprendimento automatico), che consente alle macchine di analizzare grandi quantità di dati e identificare modelli, tendenze e correlazioni al loro interno. Attraverso l'apprendimento automatico, le macchine possono acquisire conoscenze e competenze basate sui dati e utilizzarle per prendere decisioni e fare previsioni. Un'altra importante branca dell'IA è rappresentata dal *Deep Learning* (chiamato anche apprendimento profondo), che si basa su reti neurali artificiali profonde per analizzare dati complessi e svolgere compiti di riconoscimento di immagini, elaborazione del linguaggio naturale, traduzione automatica, e molto altro ancora raggiungendo risultati notevoli in molte applicazioni e dimostrando un'enorme capacità di apprendimento e adattamento.
L'IA ha anche visto lo sviluppo di altre aree di ricerca e applicazioni, come la robotica intelligente, l'elaborazione del linguaggio naturale, la visione artificiale, l'elaborazione delle immagini e l'elaborazione del suono. Queste tecnologie stanno rivoluzionando l'automazione industriale, migliorando le cure mediche, ottimizzando la gestione delle risorse e portando a nuove opportunità di sviluppo in svariati settori.
Tuttavia, essa solleva anche questioni etiche e sociali importanti, come la responsabilità delle decisioni automatizzate, la privacy dei dati, l'equità e l'impatto sull'occupazione. È fondamentale considerare tali aspetti e sviluppare linee guida e regolamenti appropriati

per garantire un utilizzo responsabile, etico e sostenibile dell'IA.
A tal proposito l'*Unione Europea* ha avanzato una proposta di regolamento (L'*AI Act* [8]) relativa all'intelligenza artificiale che è stata presentata dalla *Commissione Europea* nel 2021. L'*AI Act* è progettato per regolare l'uso e lo sviluppo di sistemi di intelligenza artificiale nell'UE al fine di garantire un ambiente sicuro, etico e trasparente per l'adozione di questa tecnologia. Alcuni dei principali punti trattati dall'AI Act sono:

- **Categorie di IA ad alto rischio**: L'*AI Act* identifica specifiche categorie di sistemi di IA considerati ad alto rischio, come quelli utilizzati nella sanità, nell'automotive, nella sicurezza e in altri settori critici. Tali sistemi saranno soggetti a requisiti e standard più rigorosi.
- **Obblighi per i fornitori di IA**: Il regolamento stabilisce obblighi per i fornitori di sistemi di IA ad alto rischio, tra cui valutazioni di conformità, documentazione tecnica, monitoraggio e segnalazione, e la necessità di nominare un rappresentante responsabile.
- **Dati e trasparenza**: L'*AI Act* sottolinea l'importanza della qualità e della provenienza dei dati utilizzati nei sistemi di IA e richiede una maggiore trasparenza nell'uso di tali dati.
- **Agenzia europea per l'intelligenza artificiale**: Il regolamento propone l'istituzione di un'agenzia europea dedicata all'IA per fornire orientamenti e supporto tecnico agli Stati membri.
- **Vietato l'uso di determinate pratiche**: L'*AI Act* vieta l'uso di pratiche considerate inaccettabili, come la creazione di sistemi di IA che manipolano le persone o sfruttano le vulnerabilità delle persone.
- **Sanzioni e penalizzazioni**: Vengono previste sanzioni significative per le violazioni del regolamento, che possono includere multe sostanziali.

Pertanto l'*AI Act* mira a garantire che l'adozione dell'intelligenza artificiale nell'UE avvenga in modo responsabile, rispettando i diritti dei cittadini e la sicurezza pubblica. È stato oggetto di discussioni e negoziazioni all'interno dell'UE e potrebbe subire modifiche prima di essere adottato definitivamente come legge.
Tuttavia, l'opacità e l'incomprensibilità degli algoritmi di intelligenza artificiale possono sollevare preoccupazioni in termini di fiducia, responsabilità e accettabilità sociale ed in risposta a queste preoccupazioni, è emersa la necessità di sviluppare metodi di spiegabilità dell'IA, noti come Explainable Artificial Intelligence (XAI).

1.2 Explainable Artificial Intelligence

Negli ultimi anni, i modelli di Machine Learning (ML) e di Deep Learning (DL) hanno ottenuto risultati straordinari in molti settori, specialmente in quelli con dati complessi come la visione artificiale e l'elaborazione del linguaggio naturale. Questo successo è stato reso possibile grazie all'aumento dei dati disponibili, al miglioramento delle prestazioni

hardware e agli algoritmi di ottimizzazione che consentono un addestramento efficiente dei modelli di ML. Tuttavia, modelli complessi come le *Reti neurali profonde* (DNN) e le *Support Vector Machine* (SVM) sono spesso considerati "scatole nere" dagli utenti a causa della loro complessità intrinseca. Questa mancanza di trasparenza sul funzionamento interno solleva preoccupazioni sulla fiducia degli utenti e sulla possibilità di pregiudizi nei dati utilizzati per addestrare i modelli.I modelli più trasparenti, come quelli lineari o basati su alberi decisionali, hanno una complessità inferiore e una logica di funzionamento più chiara. Tuttavia, potrebbero essere meno precisi o meno adatti per compiti che coinvolgono dati complessi ad alta dimensionalità. La complessità delle DNN e delle SVM può nascondere distorsioni nei dati di addestramento e rendere difficile identificare le cause dei pregiudizi. Questa situazione può avere conseguenze pericolose, specialmente quando si trattano dati sensibili o si prendono decisioni critiche, come nel campo della medicina, delle finanze o dell'assunzione di personale. Per affrontare queste problematiche, il Regolamento generale sulla protezione dei dati dell'Unione Europea ha introdotto il diritto di spiegazione. La capacità di fornire spiegazioni sulle previsioni del modello è fondamentale per garantire la fiducia degli utenti e consentire loro di comprendere le ragioni di una decisione presa da un sistema automatico. L'interpretabilità e la spiegabilità dei modelli di ML non solo influenzano la fiducia degli utenti, ma possono anche aiutare nello sviluppo e nell'ottimizzazione dei modelli stessi. Le spiegazioni possono aiutare a individuare errori di previsione e suggerire possibili miglioramenti prima dell'implementazione del modello. Inoltre, le spiegazioni possono contribuire alla comprensione scientifica dei processi di generazione dei dati. Combinando modelli basati su conoscenze teoriche con modelli numerici derivati dai dati, è possibile ottenere nuovi approfondimenti nel campo scientifico.
In conclusione, l'interpretabilità e la spiegabilità dei modelli di ML sono fondamentali per garantire fiducia, consentire la correzione di errori nei modelli e fornire intuizioni sui processi di generazione dei dati. L'uso delle spiegazioni può contribuire all'avanzamento della comprensione scientifica e alla scoperta di nuove ipotesi e proprietà dei processi di generazione dei dati.
All'interno dell'Explainable AI vi sono vari algoritmi e modelli di spiegazione che si possono riassumere nelle seguenti classi presentate da [5]:

- **Counterfactual Explanations**: Questo tipo di spiegazione coinvolge la creazione di scenari ipotetici ("controfattuali") che mostrano come il risultato di un modello sarebbe cambiato se le variabili in ingresso fossero state diverse. Ciò aiuta a comprendere il ruolo delle diverse variabili nei risultati del modello.

- **Counterfactual Explanations for Causality**: Questo tipo di spiegazione può coinvolgere il confronto di scenari controfattuali per analizzare le relazioni causali all'interno dei dati.

- **Causal Inference Models**: Gli algoritmi di inferenza causale cercano di identificare e quantificare le relazioni di causa-effetto all'interno dei dati e dei modelli. Questi modelli possono aiutare a spiegare perché un modello prende decisioni specifiche.

- **Interpretable Machine Learning Models**: Questi sono modelli di apprendimento automatico specificamente progettati per essere facilmente interpretabili. Esempi includono *alberi decisionali*, *regressione lineare* e *modelli lineari generalizzati*.
- **Feature Importance and Attribution Methods**: Questi metodi assegnano un'importanza o una "contribuzione" alle feature di input per spiegare come contribuiscono alle previsioni del modello. All'interno di questa classe si trovano a loro volta le classi **Local Interpretability Methods** e **Visual and Graphical Explanations**.
- **Local Interpretability Methods**: Questi metodi cercano di spiegare le singole previsioni di un modello, evidenziando quali feature influenzano una specifica previsione. All'interno della quale vi sono i *Surrogate Models* dove si trovano gli algoritmi *LIME*, *Anchors* e *KernelSHAP* presentati in questa tesi.
- **Global Interpretability Methods**: Questi metodi cercano di spiegare il comportamento generale di un modello, evidenziando le feature più influenti nell'intero insieme di dati. Di questa classe fanno parte il resto degli algoritmi *SHAP*.
- **Visual and Graphical Explanations**: L'utilizzo di grafici, visualizzazioni e altre rappresentazioni visive può aiutare a spiegare il comportamento di un modello in modo più intuitivo.
- **Rule-Based Explanations**: La creazione di regole logiche o "se-allora" può essere utilizzata per spiegare il ragionamento di un modello. Di questo gruppo fa parte nello specifico *Anchors*.

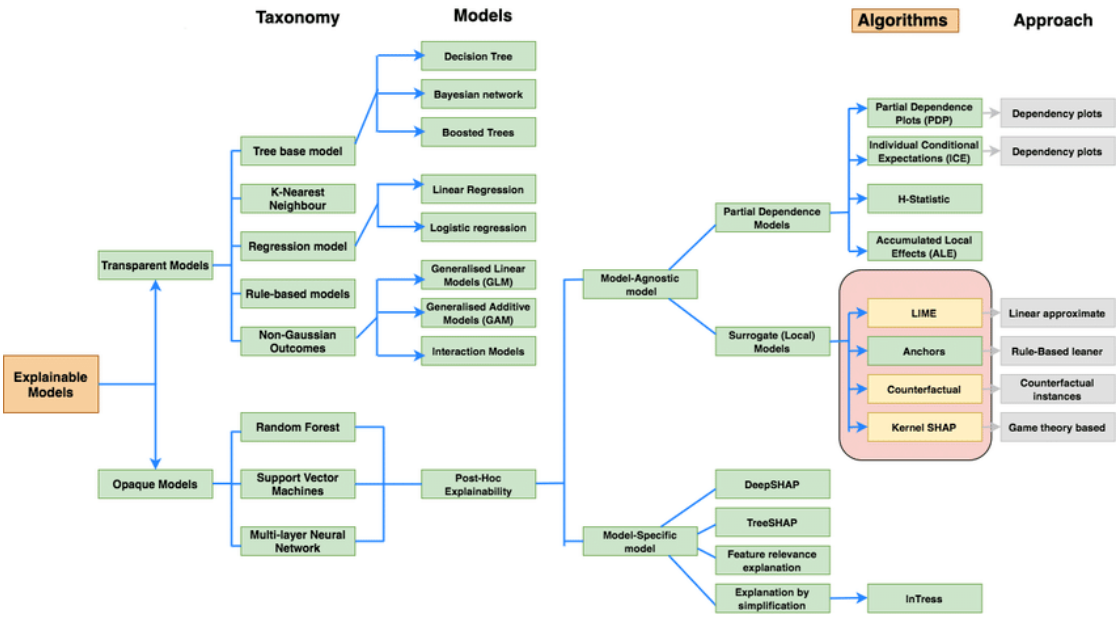


Figura 1.1: Modelli e approcci dell Explainable AI [5]

1.3 Scopo e Struttura della tesi

"Study of Shapley Value-based Explainability in Machine Learning" è stata progettata con l'obiettivo di esaminare in profondità l'importante campo dell'intelligenza artificiale (IA) e, in particolare, di affrontare la crescente necessità di comprenderne il funzionamento, al fine di sfruttarne appieno i vantaggi e mitigarne le sfide. L'IA è divenuta un pilastro fondamentale in svariati settori, ma la sua complessità e l'opacità dei modelli di machine learning hanno sollevato preoccupazioni legate alla trasparenza e all'interpretabilità.
La struttura della tesi è stata concepita in modo da affrontare una serie di obiettivi chiave: verranno presentati i principali sviluppi nell'ambito dell'IA e la sua crescente importanza nella società contemporanea. Sarà sottolineata l'importanza di comprendere il funzionamento dei modelli di machine learning per sfruttarne i vantaggi e mitigarne i difetti.
Verranno fornite le basi matematiche e concettuali necessarie per comprendere a fondo i metodi di spiegazione con un focus sulla teoria dei giochi e i concetti matematici fondamentali. Queste conoscenze costituiranno la base per la comprensione dei metodi SHAP (SHapley Additive exPlanations).
Verrà condotta un'analisi approfondita dei metodi SHAP, esaminandone la storia e il funzionamento. Saranno esplorate le ragioni che rendono questi metodi così innovativi e saranno discussi i possibili ambiti di applicazione.
Successivamente sarà esplicata la fase sperimentale della tesi, in cui verranno condotti esperimenti mirati per testare l'efficacia dei metodi SHAP in confronto ad altri metodi di spiegazione, con particolare enfasi sul metodo LIME.
I risultati saranno analizzati approfonditamente per valutare l'efficacia dei metodi in diversi contesti.
Infine saranno tratte le conclusioni derivanti dagli esperimenti, evidenziando i pregi e i difetti dei metodi SHAP e suggerendo possibili strade per migliorarli. Saranno inoltre esplorate le prospettive future di ricerca in questo campo, indicando come questi metodi possano contribuire all'evoluzione dell'interpretabilità nei modelli di machine learning.
In sintesi, la tesi "Study of Shapley Value-based Explainability in Machine Learning" mira a esaminare in dettaglio i metodi di spiegazione, concentrandosi sui metodi SHAP, e a valutarne l'applicabilità in contesti di shallow learning. Attraverso analisi rigorose e sperimentazioni si intende contribuire alla comprensione di come questi metodi possano migliorare la trasparenza e l'interpretabilità dei modelli di machine learning, con l'obiettivo di promuovere la conoscenza e l'innovazione in questo campo.
Nel contesto di questa tesi, gli esperimenti condotti sono stati ispirati e guidati dalla ricerca presentata nei seguenti articoli: [7](*Dhurhandhar*) dove vengono applicati i *"Contrastive Explanations Method"* a un dataset visuale *MNIST* utilizzando le tecniche denominata *"Finding Pertinent Negatives (PN)"* e *"Finding Pertinent Positives (PP)"* esso mira a migliorare le spiegazioni delle previsioni dei modelli, queste tecniche hanno contribuito all'elaborazione delle metriche *"keep/remove positive/negative"*, [13](*Molnar*) esamina diversi metodi di spiegazione tra cui sia gli approcci globali, come i "Global Model-Agnostic Methods," con un'attenzione particolare ai "global surrogate models," sia metodi locali, come LIME e SHAP focalizzandosi sull'interpretabilità delle reti neurali, [21](*Vilone*) tratta della crescita significativa dell'*Intelligenza Artificiale Esplicabile (XAI)* a causa dell'ampia applicazione dell'apprendimento automatico, specialmente del deep learning, che

ha portato a modelli molto accurati ma difficili da spiegare. L'articolo presenta una revisione sistematica dei metodi XAI, suddividendoli in quattro categorie principali: articoli di revisione, teorie e concetti, metodi e valutazione. Inoltre, offre una panoramica dello stato attuale della XAI e suggerisce direzioni future per la ricerca, [12] (*Scott*) si concentra sulla necessità di interpretare le previsioni ottenute da modelli basati su alberi come Gradient Boosting Machines e Random Forests, mostrando che i metodi comuni di attribuzione delle caratteristiche sono inconsistenti il che può portare a valutazioni errate delle caratteristiche. Per risolvere questo problema, gli autori applicano la teoria dei giochi per sviluppare soluzioni esatte per i valori integrando l'algoritmo in XGBoost e LightGBM. Introducono anche valori di interazione SHAP per catturare gli effetti delle interazioni tra le caratteristiche, proponendo visualizzazioni avanzate delle attribuzioni delle caratteristiche e un metodo di clustering basato su queste attribuzioni. Gli autori dimostrano una migliore comprensione umana, prestazioni più veloci, miglioramenti nel clustering e una migliore identificazione delle caratteristiche influenti.
Dal punto di vista tecnico, gli esperimenti condotti in questa tesi rivestono un'importanza fondamentale in quanto mirano a fornire una valutazione complessiva delle prestazioni degli algoritmi trattati in una vasta gamma di scenari. Questi scenari comprendono sia problemi di classificazione che di regressione, considerando dataset con e senza perturbazioni, nonché dataset con un numero variabile di feature. Inoltre, sono stati eseguiti esperimenti su dataset in cui le feature possono essere più o meno correlate, consentendo così di esaminare come le prestazioni degli algoritmi variano anche in queste situazioni specifiche.

Capitolo 2

Teoria Dei Giochi

La teoria dei giochi è una branca della matematica applicata che si occupa di studiare le decisioni e le strategie di comportamento in situazioni in cui le conseguenze dipendono dalle scelte dei diversi attori coinvolti. In particolare, la teoria dei giochi si concentra sulle interazioni tra individui razionali, che cercano di massimizzare il loro risultato, in un contesto in cui il successo dipende non solo dalle loro azioni, ma anche dalle azioni degli altri giocatori.
Essa trova applicazioni in molti campi, tra cui l'economia, la scienza politica, la biologia, la psicologia, la filosofia e, nel nostro caso, il Machine Learning. Le principali categorie di giochi sono i **giochi a somma zero**, in cui il guadagno di un giocatore corrisponde alla perdita degli altri, e i **giochi non a somma zero**, in cui i risultati possono essere positivi per tutti i giocatori o negativi per tutti.
La teoria dei giochi fornisce strumenti analitici e metodologie per studiare le interazioni strategiche tra i giocatori, la formazione di coalizioni (tema fondamentale di questa tesi), l'equilibrio delle strategie e la previsione dei risultati di gioco. La teoria dei giochi è stata utilizzata per sviluppare modelli di comportamento economico, analizzare le dinamiche dei mercati finanziari, studiare le strategie di negoziazione e risoluzione dei conflitti, e analizzare il comportamento degli animali in competizione per le risorse.
Le situazioni decisionali interattive, chiamate giochi, sono caratterizzate dalla presenza di agenti, chiamati giocatori, che prendono decisioni (strategie) che vengono modellate matematicamente utilizzando la teoria dei giochi. La teoria classica dei giochi presuppone che i giocatori agiscano in modo razionale, cioè che sappiano cosa vogliono, che il raggiungimento di tale obiettivo sia il loro scopo e che siano in grado di determinare quali strategie siano utili per raggiungere il loro obiettivo. Un'importante categoria di giochi è quella dei giochi cooperativi, ossia quei giochi in cui un insieme di giocatori, chiamato coalizione, è disposto a cooperare per raggiungere un obiettivo comune concentrando la sua attenzione su come distribuire equamente i benefici di questa cooperazione tra i giocatori. Lo Shapley Value è una delle soluzioni più note a questo problema.

2.1 Teoria delle Decisioni

La *Decision theory* (o teoria delle decisioni) è una parte della teoria dei giochi che si occupa di studiare il processo decisionale di un singolo giocatore. In particolare, la decision theory cerca di individuare i criteri che un giocatore razionale dovrebbe utilizzare per prendere una decisione in un contesto di incertezza.
In un gioco, un giocatore deve prendere una decisione sulla base delle informazioni disponibili e delle scelte degli altri giocatori. La decision theory si concentra sulle strategie di scelta di un giocatore, al fine di massimizzare il proprio risultato atteso in termini di guadagno.
Uno dei principali strumenti della decision theory è la **teoria della scelta razionale**, che fornisce un modello matematico per la presa di decisioni in un contesto di incertezza. Questo modello prevede che un giocatore razionale debba scegliere la strategia che massimizza il proprio risultato atteso, ovvero la somma pesata dei risultati possibili, ponderati per le probabilità di occorrenza.
La Teoria delle decisioni si basa sul concetto di relazione R su un insieme delle azioni A che a sua volta è un sottoinsieme di $A \times A$.
Date due azioni $a, b \in A$, tali azioni sono in relazione binaria tra loro se $(a, b) \in R$, in tal caso si scrive aRb. Pertanto, un giocare che deve effettuare una decisione, deve necessariamente scegliere tra una o più alternative nell'insieme A.
Tale relazione in A precedentemente accennata indica dunque la preferenza di un giocatore per un'azione rispetto ad un'altra e si indica con il simbolo $\succeq$, ad esempio $b \succeq a$ che si traduce quindi come "l'azione b è preferibile dal giocatore rispetto all'azione a" [10].
Si assume che tale relazione abbia anche le sequenti proprietà:

- Sia **completa**, ovvero $\forall a, b \in A$ si ha che $a \succeq b$ o $b \succeq a$ (o entrambe nel caso in cui le due azioni abbiano la stessa valenza).
- Sia **transitiva**, ovvero $\forall a, b, c \in A$ se $a \succeq b$ e $b \succeq c$ allora necessariamente $a \succeq c$ (in tal caso la relazione si dice essere una **Preferenza Debole** su A).

Definizione 1 (Problema Decisionale)**.** *Una coppia $(A, \succeq)$, dove A è l'insieme delle azioni e $\succeq$ è una preferenza debole su A viene chiamata* ***problema decisionale***

Utilizzare il concetto di Relazioni di preferenza, però, non risulta essere la migliore scelta, pertanto si preferisce invece usare il concetto di **Utility Function**.

Definizione 2 (Utility Function)**.** *Dato un problema decisionale $(A, \succeq)$, una* ***utility function*** *(funzione di utilità) che rappresenta la relazione $\succeq$ è una funzione $u : A \to \mathbb{R}$ che soddisfa*

$$u(a) \geq u(b) \iff a \succeq b, \ \forall a, b \in A$$

Ci sono alcune condizioni che garantiscono l'esistenza di una rappresentazione per un problema decisionale. Si può dimostrare che se A è numerabile, esiste una rappresentazione di $\succeq$ attraverso la *funzione di utilità*. Si può dimostrare anche che esiste una rappresentazione di $\succeq$ attraverso la funzione di utilità se $\succeq$ è *antisimmetrica* e A contiene

un sottoinsieme numerabile e denso [10]. La funzione di utilità è unica a meno di una trasformazione strettamente crescente.
Si prenda ad esempio un inseme convesso X sottoinsieme di uno spazio vettoriale reale di dimensione finita.
Un problema decisionale $(X, \succeq)$ è un problema decisionale convesso.
Si assuma inoltre che la relazione $\succeq$ sia **indipendente**, tale per cui $\forall a, b, c \in X$ e $\rho \in (0,1]$ si ha che $a \succeq b \iff \rho a + (1-\rho)c \succeq \rho b + (1-\rho)c$, e **continua**, ovvero $\forall a, b, c \in X$ con $a \succ b \succ c$ esista $\rho \in (0,1)$ tale che chiamando $\sim$ la relazione di equivalenza associata al problema si abbia che $b \sim \rho a + (1-\rho)c$ ovvero che $\forall a, b \in X$, $a \sim b \iff a \succeq b, b \succeq a$.
Si può inoltre dimostrare che esiste una **funzione di utilità lineare** $\bar{u} : X \to \mathbb{R}$ che rappresenta la relazione $\succeq$, tale per cui essa sia unica a meno di trasformazioni affini positive.
Le funzioni di utilità sono di grande importanza in quanto permettono di descrivere situazioni di interazione tra agenti diversi. Queste situazioni sono modellate tramite giochi strategici, in cui vengono specificate le strategie possibili per i giocatori, i possibili risultati e le **funzioni di payoff** per ciascun giocatore che sceglie una determinata strategia. Le funzioni di payoff per i giocatori sono strettamente correlate alle loro funzioni di utilità, che esprimono le loro preferenze sui possibili risultati. In particolare, le funzioni di utilità lineari sono di particolare interesse poiché permettono di definire giochi a strategie miste, in cui un giocatore può scegliere casualmente una **strategia (lotteria)** tra quelle disponibili.

Funzione di utilità attesa (Von Neumann-Morgenstern)

Dato un insieme delle azioni A, un **insieme di lotterie** ΔA su A, è un insieme di distribuzioni di probabilità su A con supporto finito, ovvero

$$\Delta A := \left\{ x \in [0,1]^A : |\{a \in A : x(a) > 0\}| < \infty, \sum_{a \in A} x(a) = 1 \right\}$$

Per rappresentare la relazione di preferenza nei problemi decisionali convessi, dove l'insieme delle alternative ha la forma ΔA per qualche insieme A, viene utilizzata la *funzione di utilità di von Neumann e Morgenstern.* In altre parole, questa funzione viene usata per modellare la preferenza di un individuo tra diverse alternative in un contesto di decisione, dove le alternative sono rappresentate da un insieme convesso.

Definizione 3 (Funzione di utilità attesa)**.** *Sia A un insieme di alternative e $(\Delta A, \succeq)$ un problema decisionale convesso. Una funzione $u : A \to \mathbb{R}$ che soddisfa*

$$x \succeq y \iff \sum_{a \in A} u(a)x(a) \geq \sum_{a \in A} u(a)y(a), \forall x, y \in \Delta A$$

*è chiamata **funzione di utilità di Von Neumann-Morgenstern** o funzione di utilità attesa rappresentante $\succeq$.*

La funzione di utilità di von Neumann e Morgenstern rappresenta le preferenze di un giocatore in un insieme convesso di probabilità ΔA, ma è definita sull'insieme delle alternative A. Essa esiste solo se esiste una funzione di utilità lineare che rappresenta

la debole preferenza sulle lotterie in ΔA. Tali funzioni di utilità sono anche chiamate funzioni di utilità attese, poiché assegnano un valore di utilità a ogni alternativa e il valore atteso delle lotterie viene calcolato dal giocatore per stabilire le sue preferenze sulle lotterie. Queste funzioni di utilità sono di grande importanza in quanto costituiscono le fondamenta teoriche per lo sviluppo sia della teoria dei giochi cooperativi che di quelli non cooperativi.

2.1.1 Teoria dei giochi cooperativi

La teoria dei giochi può essere suddivisa in due categorie principali: **modelli cooperativi** e **modelli non cooperativi**. Nella **teoria dei giochi cooperativi** si considerano situazioni interattive con più giocatori che possono decidere di cooperare per massimizzare il risultato sociale ottimale. L'obiettivo principale è quello di individuare come distribuire equamente i benefici derivanti dalla cooperazione. D'altro canto, nella **teoria dei giochi non cooperativi**, i giocatori agiscono simultaneamente e indipendentemente l'uno dall'altro cercando di massimizzare il proprio guadagno individuale utilizzando strategie indipendenti [10]. In questa trattazione ci concentreremo sulla teoria dei giochi cooperativi. Si denoti con N l'insieme degli n giocatori ($N = \{1,2,\dots,n\}$). Ogni sottoinsieme S di N, rappresentante un sottogruppo di giocatori, è chiamato **coalizione**.
Si denoti con $|S|$ la cardinalità della coalizione (il numero dei suoi giocatori). La coalizione formata da tutti i giocatori, che coincide con N, è chiamata **grande coalizione**. Il presupposto che i giocatori all'interno di una coalizione possano negoziare efficacemente è di fondamentale importanza in questo contesto. Ciò implica che se un cambiamento nella strategia della coalizione comporta un vantaggio per tutti i suoi membri, allora essi saranno d'accordo ad adottare tale modifica, a meno che non esista un accordo pregresso di uno o più giocatori della coalizione con i membri esterni, supponendo che quest'ultima coalizione sia quantomeno di pari efficienza. La teoria dei giochi cooperativi è un campo piuttosto complesso a causa dell'effetto significativo che l'ordine di negoziazione delle coalizioni e la scelta degli accordi vincolanti o non vincolanti hanno sui giochi di interazione [14]. Di conseguenza, ulteriori presupposti sono necessari per poter studiare in modo approfondito tali giochi. Un presupposto comune è l'esistenza di un "bene" che può essere trasferito tra i giocatori per aumentare il loro guadagno. Questo tipo di giochi viene definito "giochi di utilità trasferibile", mentre in assenza di utilità trasferibili il gioco viene definito come "gioco di utilità non trasferibile", che rappresenta la classe più generale dei giochi cooperativi.

Definizione 4 (Gioco di utilità non trasferibile)**.** ***Un gioco di utilità non trasferibile ad n giocatori (NTU-game)*** *è una coppia (N, V) in cui N rappresenta l'insieme di giocatori e V è una funzione che assegna ad ogni coalizione $S \subset N$ un insieme $V(S) \subset \mathbb{R}^S$, dove $\mathbb{R}^S$ è lo spazio euclideo di dimensione $|S|$. Per convenzione si ha che $V(\emptyset) := \{0\}$ (vettore nullo). Inoltre, per ogni $S \subset N$ non vuoto, si verifica che:*

- *$V(S) \subset \mathbb{R}^S$ è chiuso e non vuoto.*
- *$V(S)$ è esaustivo, ovvero $\forall x, y \in \mathbb{R}^S$ tale che $x \in V(S)$ e $y \leq x$ si ha che $y \in V(S)$. Inoltre, $V(\{i\}) \neq \mathbb{R}$, $\forall i \in N$, cioè $\exists v_i \in \mathbb{R}$ tale che $V(\{i\}) = (-\infty, v_i]$.*

- *L'insieme* $V(S) \cap \{y \in \mathbb{R}^S : y_i \geq v_i, \forall i \in S\}$ *è limitato.*

In teoria dei giochi coalizionali, non si descrive un gioco attraverso la specificazione dettagliata delle strategie, degli esiti e della funzione di utilità per tutti i possibili giocatori, ma invece si definisce il gioco attraverso il payoff disponibile per ogni coalizione (in forma coalizionale). Questo presuppone che ci sia un insieme di esiti che solo i giocatori di una data coalizione possono raggiungere, e che per ogni giocatore sia possibile definire una relazione di preferenza e una funzione di utilità, in modo che per ogni esito raggiunto dalla coalizione, ogni giocatore riceva un payoff che corrisponde alla sua funzione di utilità. Se inoltre si assume che l'insieme degli esiti raggiungibili dalla coalizione sia convesso, allora le funzioni di utilità dei giocatori in quella coalizione saranno funzioni di utilità di von Neumann e Morgernstern.

Definizione 5 (Allocazioni valide)**.** *Sia (N,V) un NTU-game. I vettori in* $\mathbb{R}^N$ *sono chiamate allocazioni. Le allocazioni* $x \in \mathbb{R}^N$ *sono dette valide se esiste una partizione di N denotata con* $\{S_1, ..., S_k\}$*, tale che*

$$y \in V(S_l), \forall l \in \{1, ..., k\} \text{ con } y_i = x_i, \forall i \in S_l.$$

Il principale obiettivo della teoria dei giochi cooperativi è trovare regole, dette soluzioni, per selezionare allocazioni fattibili per un gioco NTU con alcune proprietà desiderate. Queste regole devono essere appropriate per scegliere l'allocazione migliore per ogni gioco e se l'allocazione selezionata è unica, allora la soluzione è chiamata regola di allocazione [10]. Poiché la classe di giochi NTU è difficile da studiare, la letteratura si è concentrata su una classe particolare, ovvero quella dei giochi a utilità trasferibile o TU-games.

2.1.2 Giochi a utilità trasferibile

I giochi a utilità trasferibile (TU-games) rappresentano una classe di giochi cooperativi in cui i giocatori possono trasferire l'utilità tra di loro. In questi giochi le allocazioni sono determinate dalle coalizioni di giocatori, e l'obiettivo principale è dividere in modo equo i benefici della cooperazione tra i membri della coalizione.
Un'importante ipotesi dei giochi a utilità trasferibile è che, se una coalizione S impone un'allocazione $x \in V(S) \subset \mathbb{R}^S$, allora tutte le allocazioni che si ottengono trasferendo l'utilità tra i giocatori di S appartengono anch'esse a $V(S)$. Pertanto, la classe di allocazioni $V(S)$ può essere rappresentata da un unico valore, chiamato valore della coalizione S, ottenuto come il massimo guadagno ottenibile dalla coalizione scegliendo l'allocazione più conveniente tra tutte quelle disponibili in $V(S)$.

Definizione 6 (TU-game)**.** *Un gioco a utilità trasferibile (TU-game) è definito come una coppia* (N, v)*, dove N indica l'insieme dei giocatori e* $v : 2^N \to \mathbb{R}$ *è la funzione caratteristica del gioco, ovvero è definita come una mappa da* 2^N *(l'insieme di tutte le possibili coalizioni) a* $\mathbb{R}$ *(l'insieme dei numeri reali). Fissando* $v(\emptyset) := 0$*.*

Il valore di una coalizione rappresenta il beneficio che una coalizione può generare senza l'aiuto dei giocatori che non fanno parte della stessa. Talvolta v viene utilizzata anche per indicare il gioco (N, v). Da ora la notazione $v(i)$ e $v(ij)$ indicherà rispettivamente

$v(\{i\})$ e $v(\{i,j\})$. La classe di giochi a utilità trasferibile con n giocatori viene indicata con G^N. Ogni TU-game (N,v) è un NTU-game (N,V), dove $\forall S \subset N$, con $S \neq \emptyset$, $V(S) \coloneqq \{y \in \mathbb{R}^S : \sum_{i \in S} y_i \leq v(S)\}$.

Definizione 7 (Restrizione di (N,v) a S)**.** *Sia $(N,v) \in G^N$. Per ogni coalizione $S \subset N$, il TU-game (S, v_s), dove $v_s(T) \coloneqq v(T)$, $\forall T \subset S$ è chiamata restrizione di (N,v) a S.*

Definizione 8 (TU-game superadditivo)**.** *Un TU-game $v \in G^N$ è chiamato superadditivo se $\forall S,T \subset N$, con $S \cap T \neq \emptyset$, e vale che $v(S \cup T) \geq v(S) + v(T)$. L'insieme dei TU-games superadditivi con n giocatori è denotato con SG^N*

Definizione 9 (Copertura superadditiva [14])**.** *La **copertura superadditiva** di un gioco $v \in G^N$ è definita come la funzione superadditiva $w \in SG^N$ tale che per ogni coalizione $S \subseteq N$, $w(S)$ è il valore minimo possibile tale che $w(S) \geq v(S)$.*
Esso soddisfa

$$w(S) = \max \left(\sum_{i=1}^{k} v(S_j) \middle| \{S_1, ..., S_k\} \in \mathcal{P}(S) \right)$$

$\mathcal{P}(S)$ rappresenta l'insieme di tutte le partizioni di S, per tutte le coalizioni $S \subseteq N$.

In un gioco superadditivo i giocatori sono incentivati a cooperare poiché l'unione di qualsiasi coppia di coalizioni disgiunte non diminuisce il beneficio complessivo. Per questo motivo si assume che si formi la grande coalizione e che l'obiettivo sia di dividere equamente il valore totale tra i giocatori dopo il processo di contrattazione. Il risultato dell'allocazione dipende dal potere dei giocatori, ovvero dalla loro capacità di aumentare o diminuire il valore delle coalizioni attraverso la cooperazione o il rifiuto di cooperare. La funzione caratteristica descrive la struttura del potere dei giocatori nel gioco.
L'obiettivo dell'analisi dei giochi a utilità trasferibile è prevedere l'esito del processo di negoziazione tra i giocatori; si assume che si formi la grande coalizione e che il valore totale del gioco sia diviso equamente tra i giocatori in modo equo e ammissibile. L'allocazione dei pagamenti ai giocatori dipende dalla loro posizione di potere e l'obiettivo è trovare una regola di allocazione equa ed univoca. Il concetto degli Shapley Value è un approccio utilizzato per affrontare questo problema.

Capitolo 3

Lo Shapley Value

Il valore di Shapley è un concetto di soluzione nella teoria dei giochi cooperativi. È stato così chiamato in onore di Lloyd Shapley, che l'ha introdotto nel 1951 e ha vinto il Premio Nobel per l'Economia nel 2012 [18].
Per ogni gioco cooperativo il valore di Shapley assegna una distribuzione unica (tra i giocatori) di un surplus totale generato dalla coalizione di tutti i giocatori. Il valore di Shapley è caratterizzato da una serie di proprietà specifiche.

Definizione 10 (Regola di allocazione)**.** *Si consideri un TU-game con n giocatori. Una regola di allocazione è una mappa* $\phi : G^N \to \mathbb{R}^N$

Il **payoff atteso** di un giocatore i viene indicato come $\phi_i(v)$, mentre $\phi(v) = (\phi_i(v))_{i \in N}$ rappresenta l'insieme dei payoff attesi di tutti i giocatori. Le proprietà proposte da Shapley permettono di caratterizzare in modo unico una regola di allocazione.
Si enunci quanto segue.

Definizione 11 (Giocatore nullo)**.** *Sia* $v \in G^N$*. Un giocatore* $i \in N$ *che soddisfa* $v(S \cup \{i\}) - v(S) = 0$*,* $\forall S \subset N \setminus \{i\}$*, è definito giocatore nullo.*

Definizione 12 (Giocatori simmetrici)**.** *Due giocatori* $i, j \in N$ *sono detti simmetrici se* $v(S \cup \{i\}) = v(S \cup \{j\})$*,* $\forall S \subset N \setminus \{i, j\}$*.*

Sia ϕ una regola di allocazione; si assume che essa soddisfi i seguenti assiomi:

- **Linearità (LIN)**: dati $u, v \in G^N$ e un numero $p \in [0,1]$, si definisca il gioco coalizionale $pu + (1-p)v$ tale che $\forall S \subseteq N$, $(pu + (1-p)v)(S) = pu(S) + (1-p)v(S)$. Allora, per ogni giocatore $i \in N$

$$\phi_i(pu + (1-p)v) = p\phi_i(u) + (1-p)\phi_i(v) \tag{3.1}$$

- **Simmetria (SYM)**: $\forall v \in G^N$, due giocatori simmetrici $i, j \in N$ ricevono lo stesso payoff

$$\phi_i(v) = \phi_j(v) \tag{3.2}$$

- **Giocatore nullo (NPP) o Dummy player**: $\forall v \in G^N$, ogni giocatore nullo $i \in N$, che è un giocatore che non crea beneficio in nessuna coalizione, dovrebbe ricevere payoff pari a zero
$$\phi_i(v) = 0 \tag{3.3}$$

- **Efficienza (EFF)**: $\forall v \in G^N$, il valore totale della grande coalizione, $v(N)$, è allocato da ϕ a tutti i giocatori
$$\sum_{i \in N} \phi_i(v) = v(N) \tag{3.4}$$

- **Test autonomo (AT)**: Se v è una funzione di insieme subadditiva, cioè $v(S \cup T) \leq v(S) + v(T)$, allora, per ogni agente i, si ha che $\phi_i(v) \leq v(i)$. Allo stesso modo, se v è una funzione di insieme superadditiva, cioè $v(S \cup T) \geq v(S) + v(T)$, allora, per ogni agente i, si ha che $\phi_i(v) \geq v(i)$. Quindi, se la cooperazione ha esternalità positive, tutti gli agenti guadagnano (debolmente), e se ha esternalità negative, tutti gli agenti perdono (debolmente).

Shapley originariamente ha presentato tre assiomi: *assioma del carrier, simmetria* e *additività*.

- **Assioma di simmetria:** l'assioma è stato descritto in termini di permutazioni. Una **permutazione di** N è una funzione biunivoca che mappa ogni giocatore j ad esattamente un giocatore i tale che $\pi(i) = j$.
  L'assioma di simmetria afferma che, dato un gioco v e una permutazione π, il ruolo di un giocatore in v è lo stesso del giocatore $\pi(i)$ nel gioco permutato, senza considerare la sua etichetta.

- **Assioma dell'additività:** Esso sostituisce *l'assioma di linearità* e afferma che la funzione di allocazione ϕ soddisfa $\phi(u + v) = \phi(u) + \phi(v)$ $\forall u, v \in G^N$.
  Questo assioma non è motivato da alcun principio di equità, ma è una richiesta naturale per la regola di allocazione.

- **Assioma del carrier:** una coalizione R è un **carrier** di un gioco $v \in G^N$ se e solo se $v(S \cap R) = v(S)$, $\forall S \subseteq N$.
  L'assioma pertanto afferma che, per qualsiasi coalizione R che è un carrier,
$$\sum_{i \in R} \phi_i(v) = v(R)$$

Infine, **gli assiomi dell'efficienza e del giocatore nullo** sono enunciati in termini dell'**assioma del carrier** (in realtà, Shapley presenta solo l'efficienza, poiché il giocatore nullo deriva da essa).
Poiché, per definizione di carrier, si ha che $v(R) = v(R \cap N) = v(N)$, ovvero N è un *carrier*, questo assioma **implica l'efficienza**. Inoltre, se R è un carrier di v, i giocatori che non appartengono a R sono **giocatori nulli** (chiamati anche **giocatori dummy**), poiché non modificano il valore di alcuna coalizione. Infatti, $\forall S \subseteq N$ e $i \notin R$ si ha che

$$v(S \cup \{i\}) = v((S \cup \{i\}) \cap R) = v((S \cap R) \cup (\{i\} \cap R)) = v(S \cap R) = v(S)$$

dove la prima e la quarta uguaglianza sono dovute al fatto che R è un carrier e la terza al fatto che $i \notin R$.
Inoltre, dalla definizione di carrier, si ottiene che $v(i) = v(\{i\} \cap R) = v(\emptyset) = 0$. E si ha che se i è un giocatore nullo, $\phi_i(v) = 0$.
Secondo la **condizione di linearità** lo Shapley value deve essere lo stesso sia prima che dopo la risoluzione dell'incertezza [14].
Questo può essere applicato ad esempio in un gioco in cui i giocatori decidono di giocare una mossa con una certa probabilità a seconda dell'esito di un evento casuale. Se i giocatori negoziano prima dell'evento la situazione può essere rappresentata come una combinazione pesata di due possibili scelte mentre, se negoziano dopo, il guadagno atteso per ciascun giocatore dipenderà dalle probabilità di gioco associate a ogni scelta possibile. La condizione di linearità richiede che la regola di assegnazione dello Shapley value non dipenda dal momento in cui i giocatori negoziano, ma sia costante indipendentemente dalla presenza di incertezza.

Definizione 13 (Shapley value)**.** *Lo Shapley value ϕ è definito come il vettore delle allocazioni $\phi(v)$, con componenti $\phi_i(v), \forall i \in N$, dato da*

$$\phi_i(v) \coloneqq \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!}(v(S \cup \{i\}) - v(S)) \tag{3.5}$$

$\forall v \in G^N, \forall i \in N$

Il valore di Shapley è un metodo che assegna a ogni giocatore un valore che riflette la sua contribuzione marginale $(v(S \cup \{i\}) - v(S))$ alle possibili coalizioni, con un peso dato dalla probabilità che quella coalizione si formi. Questo valore può essere interpretato come l'aspettativa del contributo marginale di un giocatore alla coalizione, supponendo che i giocatori entrino uno alla volta in una stanza in cui si formerà la grande coalizione. Il numero di possibili ordini in cui i giocatori possono entrare nella stanza è pari a $n!$, dove n è il numero totale di giocatori, e ogni ordine ha la stessa probabilità. La probabilità che la coalizione S si formi quando il giocatore i entra nella stanza dipende dal numero di giocatori che già si sono uniti alla coalizione S, quindi è data dal rapporto tra il numero di permutazioni possibili dei giocatori rimanenti nella coda e il numero totale di permutazioni $\frac{|S|!(n-|S|-1)!}{n!}$. Il valore di Shapley è quindi calcolato come la media ponderata del contributo marginale del giocatore a tutte le possibili coalizioni con i pesi corrispondenti alle probabilità di formazione della coalizione.
Prima di procedere con la dimostrazione che il valore di Shapley soddisfa gli assiomi dati e che rappresenta l'unica regola di allocazione, è necessario introdurre il concetto di **Unanimity game**. Tale concetto sarà di fondamentale importanza per la dimostrazione del teorema.

Definizione 14 (Unanimity game)**.** *Si consideri un gioco in G^N e una coalizione $S \subset N$. Il gioco (S, w^s) definito come*

$$w^s(T) \coloneqq \begin{cases} 1 & \text{se } S \subset T \\ 0 & \text{altrimenti} \end{cases} \tag{3.6}$$

$\forall T \subset N$, *(la coalizione* T *vale* 1 *se* T *contiene tutti i giocatori in* S, 0 *altrimenti) è chiamato* ***unanimity game*** *della coalizione* S.

Ora si può dimostrare il seguente teorema

Teorema 1 (Unicità dello Shapley value)**.** *Lo Shapley value definito in (4.2) è l'unica regola di allocazione in* G^N *che soddisfa gli assiomi di efficienza, giocatore nullo, linearità e simmetria.*

Dimostrazione. Per dimostrare il teorema, prima si dimostra che la formula del valore di Shapley nella formula (4.2) soddisfa gli assiomi. Successivamente, si dimostra l'unicità della soluzione.
Sia ϕ definita come nella formula (4.2). Si dimostri che ϕ soddisfa l'**assioma di simmetria**.
Nella formula, ciò che è importante per una coalizione è se essa contiene i e il numero di giocatori che contiene, ovvero $|S|$.
Si consideri la somma nella formula (4.2): si può distinguere per $S \subset N\{i,j\}$ e $S \cup \{j\} \subset N\{i\}$. $\forall v \in G^N$ e $\forall i,j \in N$, con i e j giocatori simmetrici, se $S \subset N\{i,j\}$, $v(S \cup \{j\}) = v(S \cup \{i\})$, il che implica che il contributo marginale di i a S è uguale al contributo marginale di j. Se si considera un insieme del tipo $S \cup \{j\} \subset N\{i\}$, si ha $v((S \cup \{j\}) \cup \{i\}) - v(S \cup \{j\}) = v((S \cup \{i\}) \cup \{j\}) - v(S \cup \{i\})$, poiché $S \subset N\{i,j\}$ e i,j sono simmetrici. Pertanto, il contributo marginale di i all'insieme $S \cup \{j\}$ è uguale al contributo marginale di j all'insieme $S \cup \{i\}$. Si può quindi concludere che $\phi_i(v) = \phi_j(v)$.
Si provi adesso che lo Shapley value soddisfa l'**assioma di linearità**.
Si deve mostrare che per $u, v \in G^N$ e per un qualche $p \in [0,1]$ vale (3.1). Si ha

$$\begin{aligned}
\phi_i(pu + (1-p)v) &= \sum_{S \subseteq N\{i\}} \frac{|S|!(n-|S|-1)!}{n!}((pu + (1-p)v)(S \cup \{i\}) - (pu + (1-p)v)(S)) \\
&= \sum_{S \subseteq N\{i\}} \frac{|S|!(n-|S|-1)!}{n!}[pu(S \cup \{i\}) + (1-p)v(S \cup \{i\}) - pu(S) + \\
&\qquad + (1-p)v(S)] \\
&= p\phi_i(u) + (1-p)\phi_j(v)
\end{aligned}$$

Si mostri ora che che ϕ soddisfa l'**assioma del giocatore nullo**.
Per ogni $v \in G^N$, se i è un giocatore nullo, $v(S \cup \{i\}) - v(S), \forall S \subseteq N \setminus \{i\}$.
Pertanto, $\phi_i(v) = 0$.

Si provi che ϕ soddisfa l'**assioma di efficienza**.
Per ogni $v \in G^N$, bisogna dimostrare che

$$v(N) = \sum_{i \in N} \phi_i(v) = \sum_{i \in N} \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n-|S|-1)!}{n!}(v(S \cup \{i\}) - v(S))$$

$v(N)$ nella seconda somma appare solo una volta per ogni giocatore, con segno positivo, per $S = N \setminus \{i\}$. Infatti, $\forall i \in N$ e $S = N \setminus \{i\}$, $v(S \cup \{i\}) - v(S) = v((N \setminus \{i\}) \cup \{i\}) -$

$v(N \setminus \{i\}) = v(N) - v(N \setminus \{i\})$ con coefficiente

$$\frac{|S|!(n-|S|-1)!}{n!} = \frac{(n-1)!(n-(n-1)-1)!}{n(n-1)!} = \frac{1}{n}$$

Quindi, sommando il contributo di tutti i giocatori, nella doppia somma, $v(N)$ compare con coefficiente 1. Bisogna mostrare che i termini per tutte le altre coalizioni si annullano. Per ciascuna delle altre coalizioni $S \subset N\{i\}$, i valori $v(S)$ compaiono con segno positivo e negativo. Se $|S| = k$, il valore della coalizione S appare k volte con segno positivo, una per ogni giocatore in S, con coefficiente $\frac{(k-1)!(n-k)!}{n!}$ (qui si considera che $S = S' \cup \{j\}$, per qualche $S' \subset N$ e ogni giocatore $j \in S$, quindi il coefficiente terrà conto di $|S'| = k-1$). Il valore della coalizione S appare anche $n-k$ volte con segno negativo, uno per ogni giocatore non in S, con coefficiente $\frac{k!(n-k-1)!}{n!}$. Così si ha

$$\frac{(k-1)!(n-k)!}{n!}kv(S) - \frac{k!(n-k-1)!}{n!}(n-k)v(S) = 0$$

e la contribuzione di ogni coalizione si cancella, ad eccezione della contribuzione delle grandi coalizioni. Pertanto si ha

$$\sum_{i \in N} \phi_i(v) = \sum_{i \in N} \frac{1}{n} v(N) = \frac{n}{n} v(N) = v(N)$$

soddisfando così l'assioma di efficienza.
Quindi, si è dimostrato che lo Shapley value soddisfa i quattro assiomi. Ora bisogna dimostrare che questo valore è l'unico. Per prima cosa, si dimostri che ϕ è una **trasformazione lineare**. Sia (N, z) il gioco che assegna un valore di 0 a ogni coalizione, cioè $z(S) = 0, \forall S \subseteq N$. Per l'assioma del giocatore nullo, $\phi_i(z) = 0, \forall i \in N$. Dall'assioma di linearità, dati $u, v \in G^N$, $\phi(pu + (1-p)v) = p\phi_i(u) + (1-p)\phi_i(v)$, e per $v = z$, si ha $\phi_i(pu) = p\phi_i(u), \forall i \in N$. Quindi, ϕ è una trasformazione lineare da $\mathbb{R}^{L(N)}$ a $\mathbb{R}$, cioè $\phi : \mathbb{R}^{L(N)} \to \mathbb{R}^N$, dove $L(N) = S : S \subseteq N, S \neq \emptyset$ è l'insieme di tutte le coalizioni non vuote dei giocatori in N. Poiché $|L(N)| = 2^N - 1, \mathbb{R}^{|L(N)|}$ è uno spazio vettoriale di dimensione $(2N-1)$. Poiché ogni $v \in G^N$ può essere visto come il vettore $v(S)_{S \subseteq L(N)} \in \mathbb{R}^{2^N-1}$, allora G^N può essere identificato con $\mathbb{R}^{2^N-1}$. Si dimostri ora che l'insieme dei giochi di unanimità, $U(N) \coloneqq \{w^S : S \in L(N)\}$, è una base per tale spazio vettoriale, cioè vorremmo mostrare che $U(N)$ è un insieme di vettori linearmente indipendenti.
Dagli assiomi di efficienza e giocatore nullo si ottiene

$$\sum_{i \in S} \phi_i(w^S) = 1 \text{ e } \phi_j(w^S) = 0, \forall j \notin S$$

L'**assioma di simmetria** afferma che tutti i giocatori in S devono ottenere lo stesso payoff, quindi $\phi_i(w^S) = \frac{1}{|S|}, \forall i \in S$. Ci sono $2^N - 1$ giochi di unanimità in $U(N)$, uno per ogni coalizione non vuota. Inoltre, questi giochi sono linearmente indipendenti nello spazio vettoriale $\mathbb{R}^{L(N)}$. Infatti, per dimostrarlo, bisogna mostrare che

$$\sum_{s \in L(N)} \alpha_S w^S = 0 \implies \alpha_S = 0$$

avendo $\{\alpha_S\}_{S\in L(N)} \subset \mathbb{R}$. Se quanto precede non è vero, sia T qualsiasi coalizione di dimensione minima tale che $\alpha_T \neq 0$, ovvero si assuma che non esiste alcuna $\hat{T} \subsetneq T$ tale che $\alpha_{\hat{T}} \neq 0$. Allora, si ottiene

$$0 = \sum_{S\in L(N)} \alpha_S w^S(T) = \sum_{S\subseteq T, S\neq\emptyset} \alpha_S = \alpha_T \neq 0$$

che è un assurdo.
Quindi, $U(N)$ è una base per $\mathbb{R}^{L(N)}$ perché è linearmente indipendente e contiene tanti vettori quanti la dimensione dello spazio. Ora, una mappa lineare è univocamente determinata da ciò che fa su una base del dominio. Pertanto, esiste una *mappatura lineare unica* $\phi : \mathbb{R}^{L(N)} \to \mathbb{R}^N$ che soddisfa

$$\phi_i(w^S) = \begin{cases} \frac{1}{|S|} & i \in S \\ 0 & i \notin S \end{cases}$$

per ogni gioco w^S, perchè $U(N)$ è una base di G^N. □

Si può dimostrare che i quattro assiomi dati nella caratterizzazione del valore di Shapley sono tutti ugualmente importanti e garantiscono l'unicità dell'allocazione. Se se ne rimuove uno, si può trovare una regola di allocazione diversa dal valore di Shapley che soddisfa i restanti tre assiomi.
La nozione di valore di Shapley può essere considerata una misura dell'utilità dei giocatori. Inizialmente non è facile ricondurre la teoria dell'utilità alla definizione del valore di Shapley. Tuttavia si può dimostrare che i valori di Shapley possono essere interpretati come una *funzione di utilità di von Neumann-Morgenstern* [10].

3.0.1 Calcolo dello Shapley Value

Il concetto di valore di Shapley è stato introdotto come unico criterio di allocazione che garantisce l'equità nella condivisione del valore della grande coalizione tra tutti i giocatori di un gioco coalizionale. Lo Shapley value è anche una metrica che misura il potere di ogni giocatore all'interno del gioco. Per calcolare i valori di Shapley, viene utilizzata una formulazione che consiste nella soluzione di un problema di minimizzazione ai minimi quadrati. Questo approccio è vantaggioso poiché semplifica il calcolo numerico degli Shapley value, che altrimenti sarebbe molto complesso. Il metodo chiamato **KernelSHAP** utilizza questa formulazione per calcolare gli Shapley value attraverso la regressione lineare. Inoltre, la formulazione degli Shapley value come valori probabilistici può essere utilizzata come punto di partenza per un altro metodo di approssimazione numerica chiamato campionamento.
Per ogni vettore di pagamento $x \in \mathbb{R}^N$ e per ogni coalizione S, la differenza
$e(S, x) := v(S) - x(S)$, dove $x(S) := \sum_{i\in S} x_i$, è chiamata **eccedenza di** S **rispetto a** x [17]. A seconda del segno, può essere interpretata come la **perdita totale** (segno negativo) o il **guadagno totale** (segno positivo) sostenuto dai giocatori in S se si allontanano da S. Si può dimostrare che il valore di Shapley può essere definito anche come la soluzione unica a un problema di minimizzazione che coinvolge il vettore di eccedenza.

Una funzione di pesi coalizionali su N è una mappa che associa un numero reale $m(S)$ ad ogni coalizione non vuota S. Questo peso può essere interpretato come la probabilità che S si formi. Si suppone che m sia positiva e simmetrica, nel senso che lo stesso peso viene assegnato alle coalizioni con la stessa cardinalità. Per ogni funzione di peso, si consideri il problema di minimizzazione: trovare

$$\min_{x\in\mathbb{R}^n} \sum_{\emptyset\neq S\subseteq N} m(S)(e(S,x)-\bar{e}(v,x))^2 \text{ tale che } v(N)=\sum_{i\in N} x_i \tag{3.7}$$

con $\bar{e}(v,x)=\frac{1}{2^n-1}\sum_{\emptyset\neq S\subseteq N} e(S,x)$.
Si può dimostrare che l'eccedenza media in x è la stessa per ogni vettore di pagamento efficiente, quindi $\bar{e}(v) := \bar{e}(v,x)$, per qualsiasi vettore di pagamento efficiente x. Si è osservato che la soluzione ottimale rimane invariata se viene sostituito qualsiasi numero reale k al posto di $\bar{e}(v)$. Di fatti

$$\sum_{\emptyset\neq S\subseteq N} m(S)(e(S,x)-k)^2 = \sum_{\emptyset\neq S\subseteq N} m(S)((e(S,x))^2+k^2-2ke(s,x))$$

Ora,

$$\sum_{\emptyset\neq S\subseteq N} m(S)e(S,x) = \sum_{\emptyset\neq S\subseteq N} m(S)(v(S)-x(S))$$

Dato che x è efficiente

$$\sum_{\emptyset\neq S\subseteq N} m(S)e(S) = \sum_{\emptyset\neq S\subseteq N} m(S)\sum_{T:t=s} x(T) = \sum_{\emptyset\neq S\subseteq N} m(S)\binom{n-1}{s-1}v(N) \tag{3.8}$$

Pertanto, la funzione obbietivo è uguale a (3.7) a meno di una costante nel set dei punti ammissibili.
Per $k=0$, la soluzione ottimale di (3.7) è quella del problema: trova

$$\min_{x\in\mathbb{R}^n} \sum_{\emptyset\neq S\subseteq N} m(S)(v(S)-x(S))^2 \text{ tale che } v(N)=\sum_{i\in N} x_i \tag{3.9}$$

Definizione 15 (Valore ai minimi quadrati)**.** *Sia $\psi : G^N \to \mathbb{R}^n$ un valore tale che $\forall v \in G^N$, $\psi(v)$ è la soluzione di (3.7) per una certa funzione peso m. ψ è quindi definita come* ***valore ai minimi quadrati****, o valore della famiglia dei minimi quadrati. (ovvero la famiglia di valori simmetrici ottenuti minimizzando (3.7). Un valore del minimo quadrato è la soluzione ottimale di (3.9).)*

Ci si concentrerà ora sui problemi nella forma (3.9). Questo problema fa parte di un quadro generale di problemi di approssimazione ai minimi quadrati [9].

Esempio 1. *Sia $A \in \mathbb{R}^{m\times k}$ una matrice, $b : \mathbb{R}^k \to \mathbb{R}^m$ e $c : \mathbb{R}^k \to \mathbb{R}^k$ due mappe lineari, per k, m interi con $k, m \geq 1$. Per ogni $v \in \mathbb{R}^k$, sia $\hat{v} = \hat{v}(A, b, c)$ la soluzione del problema di ottimizzazione quadratico*

$$\min_{Ax=b(v)} \|c(v)-x\|_Q^2 \tag{3.10}$$

dove $\|\cdot\| = \sqrt{< x|x >_Q}$ *è una* $Q-$*norm su* $\mathbb{R}^k$ *associata al prodotto interno* $< x|y >_Q = x^TQy = \sum_{j-1}^{k}\sum_{i=1}^{k} q_{ij}x_iy_j, \forall x, y \in \mathbb{R}^k$, *con* $Q = [q_{ij}]$ *una matrice* $(k \times k)$ *arbitraria simmetrica definita positiva. Si assuma che* $Ax = b(v)$ *ammette almeno una soluzione. Pertanto l'unica miglior approssimazione di* $c(v)$ *nello spazio delle soluzioni* $Ax = b(v)$ *rispetto alla norma* $\|\cdot\|_Q$ *è dato da* $\hat{v}$.

Lemma 1. $v \to \hat{v}$ *è un operatore lineare ben definito se* $Ax = b(v)$ *ammette una soluzione* $\forall v \in \mathbb{R}^k$ *e* $c : \mathbb{R}^k \to \mathbb{R}^k$ *è lineare.*

Dimostrazione. Il problema è equivalente a:

$$\min_{Ax=b(v)} \frac{1}{2}x^TQx - c'(v)^Tx \tag{3.11}$$

con $c' = Qc$. Dato che Q è definita positiva, il fatto che l'unica soluzione ottimale di (3.11) è rappresentata da x è equivalente all'esistenza di un vettore y che soddisfa le condizioni **KKT** associate

$$\begin{aligned} Qx + A^Ty &= c'(v) \\ Ax &= b(v) \end{aligned}$$

Dato che b e c' sono lineari in v, ne segue che anche le soluzioni ottimali di (3.10) sono funzioni linari di v □

Tornando al problema (3.9), questo può essere definito come nel caso precedente se si considera c come l'operatore di identità, Q una matrice diagonale e il vincolo $Ax = b(v)$ dato da $\sum_{i\in N} x(\{i\}) = v(N)$ e $x(S) = \sum_{i\in S} x(\{i\}), \forall S \subseteq N$.
pertanto, dal Lemma, una soluzione di (3.9) è lineare in v, quinsdi è un valore lineare di G^N.
Il valore ψ per G^N può essere visto come un gioco additivo che assegna il valore $\psi_S(v)$ ad ogni insieme S, con $\psi_S(v) = \sum_{i\in S}\psi_i(v)$, per ogni $S \subseteq N$. D'altra parte, ogni gioco additivo v deriva da un vettore $\psi \in \mathbb{R}^N$ tale che $v(S) = \sum_{i\in S}\psi_i$, per ogni $S \subseteq N$. Quindi lo spazio dei giochi additivi è isomorfo a $\mathbb{R}^n$. Pertanto, i valori possono essere visti come giochi additivi.

Il problema (3.9) consiste nel trovare la migliore approssimazione ai minimi quadrati (pesata da m) di un gioco v da parte di un gioco additivo x, sotto il vincolo di efficienza $x(N) = v(N)$. Quindi, possiamo dire che le approssimazioni ai minimi quadrati di giochi mediante giochi additivi generano valori che sono lineari su G^N.

È stata proposta una caratterizzazione assiomatica [18] dei valori della famiglia dei minimi quadrati, utilizzando gli assiomi di efficienza, linearità, simmetria, gioco inessenziale e monotonicità coalizionale. Gli assiomi di efficienza, linearità e simmetria sono gli stessi descritti nella sezione precedente. L'**assioma del gioco inessenziale** richiede che per qualsiasi gioco additivo v e qualsiasi $i \in N, \psi_i(v) = v(i)$, mentre l'assioma di monotonicità coalizionale richiede che $\forall v, w \in G^N$ tali che $v(S) > w(S)$ per qualche S e $v(T) = w(T)$,

$\forall T \neq S$, si ha $\psi_i(S) \geq \psi_i(w), \forall i \in S$. Si può dimostrare che i valori ψ che soddisfano l'efficienza, la linearità e la simmetria possono essere espressi come

$$\psi_i(v) = \frac{v(N)}{n} + \sum_{S:i\in S\neq N} p_s \frac{v(S)}{s} - \sum_{S:i\notin S} p_s \frac{v(S)}{n-s} \forall i \in N, \forall v \in G^N \tag{3.12}$$

per un certo p_s, per $s = 1, \ldots, n-1$. Inoltre, i valori della forma precedente soddisfano efficienza, linearità e simmetria. Quindi, l'equazione precedente rappresenta una caratterizzazione dei valori che verificano i tre assiomi citati. Con questo risultato, si può dimostrare che se ψ soddisfa anche l'assioma del gioco inessenziale e la monotonicità coalizionale, allora ψ è un valore ai minimi quadrati. Infatti, si può dimostrare che:

Teorema 2. *Un valore $\psi : G^N \to \mathbb{R}^n$ è un valore ai minimi quadrati se e solo se soddisfa gli assiomi di efficienza, linearità, gioco inessenziale e monotonicità coalizionale.*

Questa caratterizzazione della soluzione del valore ai minimi quadrati permette di identificare il valore di Shapley come membro della famiglia dei minimi quadrati. Infatti, il valore di Shapley soddisfa tutte le proprietà del teorema precedente, quindi appartiene alla famiglia dei minimi quadrati [9]. La funzione di peso utilizzata è:

$$m(S) = \frac{1}{n-1}\binom{n-2}{s-1}^{-1}.$$

Sia C lo spazio di tutti i giochi additivi su N. C ha dimensione n. Sia W una matrice diagonale $n \times n$ i cui elementi sono $w_{jj} = m(S) = \alpha_S$, dove $\alpha_S \in \mathbb{R}$ sono i pesi del problema (3.9). Se $\alpha_S > 0$ per ogni $S \subseteq N$, allora W è una matrice definita positiva e possiamo riscrivere il problema (3.9) come in (3.10):

$$\min_{x\in\mathcal{C}} \|v - x\|_W^2 \tag{3.13}$$

dove $x \in \mathcal{C}$, così che soddisfi $v(N) = \sum_{i\in N} x_i$. La precedente è equivalente a

$$\min_{Ax=b} (v-x)^T W (v-x)$$

e così dal Lemma, il precedente problema di minimizzazione ha una soluzione lineare.
Si può dimostrare che lo Shapley value appartiene a questo caso introducendo il valore probabilistico [22].

Definizione 16 (Valore di probabilità)**.** *Fissato un giocatore $i \in N$, sia $\{p_S^i : S \subset N\backslash\{i\}\}$ una distribuzione di probabilità sulla collezione di coalizioni che con contengono i. Un valore ϕ_i, per i su G^N è un valore di probabilità se $\forall v \in G^N$*

$$\phi_i(v) = \sum_{S\subset N\backslash\{i\}} p_S^i [v(S\cup\{i\}) - v(S)] \tag{3.14}$$

Definizione 17 (Valore cardinale-probabilistico)**.** *Un valore di probabilità tale che i coefficienti $\{p_S^i : S \subset N\backslash\{i\}\}$ per tutti gli $i \in N$, dipende solo dalle cardinalità delle coalizioni*

$\{i\}$, S e N, ovvero ci sono n numeri non negativi $\{p_s(n)\}_{s=0,\dots,n-1}$ tale che $p_S^i = p_s(n)$, per ogni $i \in N$ e $S \subseteq N \setminus \{i\}$, e

$$\sum_{s=0}^{n-1} \binom{n-1}{s} p_s(n) = 1$$

è chiamato ***valore cardinale-probabilistico***

Si definisca un funzionale lineare $\partial_i : G^N \times N \to \mathbb{R}$ tale che $\partial_i^v(S) = v(S \cup \{i\}) - v(S)$, ovvero ∂_i per $i \in N$ associa a un gioco v e a una coalizione S la contribuzione marginale di $i \in N$ a S rispetto a v. Indicando con $\mu_i \in \mathbb{R}$ una stima della contribuzione marginale di $i \in N$, si definisce la funzione di deviazione $\sigma : \mathbb{R} \to \mathbb{R}$ tale che

$$\sigma(\mu_i) = \sqrt{\sum_{S \subset N \setminus \{i\}} p_S (\partial_i^v(S) - \mu_i)^2}$$

L'unico minimizzatore di questa funzione, che può anche essere dedotto dalle condizioni KKT del problema dei minimi quadrati

$$\min_{\mu \in \mathbb{R}} \sum_{S \subset N \setminus \{i\}} p_S^i (\partial_i^v(S) - \mu_i)^2 \tag{3.15}$$

è

$$\mu = \mathbb{E}(\partial_i^v) = \sum_{S \subset N \setminus \{i\}} p_S^i \partial_i^v(S)$$

ovvero la contribuzione marginale attesa di $i \in N$ rispetto a v.
Il precedente è un caso speciale del problema (3.10), dove $c(v) = \partial_i^v$ e Q è la matrice diagonale con p_S nella diagonale. Nel caso dello Shapley value, si fa una supposizione speciale sulla distribuzione di p_S. Le cardinalità delle coalizioni sono distribuite uniformemente (la coalizione a cui il giocatore i si unisce è ugualmente probabile che sia di qualsiasi dimensione s) e tutte le coalizioni di dimensione s sono ugualmente probabili (le coalizioni della stessa cardinalità sono distribuite uniformemente), ovvero:

$$p_S = \frac{1}{n} \binom{n-1}{s}^{-1} = \frac{s!(n-s-1)!}{n!} \quad \forall S \subset N \setminus \{i\} \tag{3.16}$$

dove $s = |S|$. Pertanto, con questi valori di p_S, lo Shapley Value è

$$\phi_i(v) = \mathbb{E}(\partial_i^v) = \sum_{S \subset N \setminus \{i\}} p_S \partial_i^v(S) = \sum_{S \subset N \setminus \{i\}} p_S (v(S \cup \{i\}) - v(S)) \tag{3.17}$$

Che è poi la soluzione unica al problema dei minimi quadrati (equivalentemente al problema (3.9)), con i pesi specificati dall'Equazione (3.16).

Capitolo 4

Metodi e algoritmi

I metodi di spiegazione basati sulla teoria dei giochi sono metodi indipendenti dal modello utilizzati per spiegare le predizioni fatte da un algoritmo di learning. Negli ultimi anni hanno attirato molta attenzione poiché sono considerati sempre più i soli metodi basati su una solida base teorica, in quanto si basano sulle fondamenta della teoria dei giochi cooperativi.
In un gioco coalizionale, un gruppo di giocatori collabora per raggiungere un obiettivo comune e il valore di Shapley viene utilizzato per condividere i benefici ottenuti dalla loro cooperazione. Questo approccio viene applicato al machine learning per ottenere una spiegazione della predizione effettuata dal modello in termini di importanza delle sue caratteristiche di input, le quali assumono il ruolo dei giocatori. Il valore di Shapley viene utilizzato per misurare l'importanza delle diverse caratteristiche che hanno contribuito alla predizione, ad esempio nel caso di immagini i pixel rappresentano i giocatori e la previsione restituita dal modello o l'etichetta del classificatore rappresentano il payoff.
Si consideri un gioco coalizionale con n giocatori e sia N la grande coalizione, $N = \{1, .., n\}$ e $S \subseteq N$ rappresente una coalizione. Sia v la funzione caratteristica del gioco. Lo Shapley value è definito come un modo per suddividere il valore totale $v(N)$ tra i giocatori. Il payoff allocato al giocatore i con $i = \{1, ..., n\}$ dallo Shapley value è

$$\phi_i(v) \coloneqq \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!}(v(S \cup \{i\}) - v(S))$$

Lo Shapley value è l'unico valore che soddisfa gli assiomi di linearità, simmetria, giocatore nullo ed efficienza, ed assicura che i payoff allocati con lo Shapley value sono giusti e unici.

4.1 Metodi di attribuzione

Si assuma che il modello da spiegare sia una funzione f dipendente da n feature di input $x = (x_1, ..., x_n)$. $F(x)$ rappresenta la previsione effettuata dal modello utilizzando l'input x. Nel contesto dell'esplicazione della previsione, le feature assumono il ruolo dei giocatori e la previsione assume il ruolo del valore della grande coalizione, ovvero $v(N) = f(x)$. La coalizione S, per ogni $S \subseteq N$, determina quindi un sottoinsieme di feature $x_S = \{x_i\}_{i \in S}$.

Si ipotizza che g rappresenti il metodo di spiegazione utilizzato per spiegare f e che g dipenda da un input semplificato x_0, tale che $x = h_x(x_0)$, dove h_x è una mappa. Nel caso di immagini h_x può essere una funzione che mappa un vettore binario di input interpretabili x_0 nello spazio di input originale x, ovvero mappa l'insieme di pixel in 1 o 0, dove 1 indica che il pixel mantiene il suo valore [11]. Inoltre g spiega f localmente, $g(z') \approx f(x')$ se $z' \approx x'$

Definizione 18 (Metodo additivo di attribuzione delle feature)**.** *Un metodo di spiegazione della forma*

$$g(z') = \phi_0 + \sum_{i=1}^{n} \phi_i z'_i \tag{4.1}$$

dove $z' \in \{0,1\}^n$, n è il numero di feature in input, e $\phi_i \in \mathbb{R}$ è chiamato ***metodo additivo di attribuzione delle feature****.*

Il metodo di attribuzione delle feature additivo utilizza una combinazione lineare delle variabili binarie, dove ogni feature è associata ad un coefficiente di attribuzione. In questo modo, la somma di tali effetti rappresenta un'approssimazione della previsione originale $f(x)$ del modello.

4.2 SHAP (Spiegazione Additiva di Shapley)

Un'interessante proprietà dei metodi di attribuzione delle feature additivo è che essi hanno una soluzione unica che soddisfa gli assiomi dei valori di Shapley della teoria dei giochi coalizionali, se la funzione caratteristica viene definita come

$$v(S) = \mathbb{E}[f(x)|x_S = x^*_S] \tag{4.2}$$

per ogni coalizione $S \subseteq N$, dove x^*_S indica il valore delle caratteristiche in S che massimizza la previsione del modello f. Nella formula precedente, la funzione caratteristica è definita come il valore atteso dell'output del modello condizionato ai valori delle feature x_S, ovvero il valore dell'output restituito dal modello predittivo se sono noti solo i valori delle feature $\{x_i : i \in S\}$. Il valore medio globale della previsione viene indicato come $\phi_0 = \mathbb{E}[f(x)]$. Si può inoltre dimostrare che esiste solo un metodo di attribuzione delle feature additivo che soddisfa gli assiomi di efficienza, simmetria, presenza di un giocatore fittizio e linearità.
L'algoritmo additivo utilizzato per l'attribuzione delle feature è noto come Spiegazione Additiva di Shapley o SHAP [11], e i coefficienti ϕ_i dell'Eq. (4.1) rappresentano i valori di Shapley delle feature di input.
Vale pertanto il seguente teorema:

Teorema 3. *Si assuma che la mappatura semplificata dell'input $h_x(x') = x_S$, dove x_S indica un vettore di feature che ha come elementi x_i pari a 0 se $i \notin S$.* ***SHAP*** *è l'unica misura dell'importanza delle feature che unifica la classe dei metodi di attribuzione delle feature additivo definendo $\phi_0 = \mathbb{E}[f(x)]$, si ha inoltre che*

$$\phi_i(f) = \sum_{s \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} (v(S \cup \{i\}) - v(S)) \tag{4.3}$$

dove $v(S) = \mathbb{E}[f(x)|x_S]$*, con* $|S|$ *che indica il numero di indici non nulli in* x*.*

Pertanto SHAP spiega la differenza tra la previsione e la previsione media globale [2], che rappresenta la previsione che si avrebbe se non si conoscessero le caratteristiche dell'output corrente. Gli Shapley value nel contesto di un gioco coalizionale soddisfano i quattro assiomi: linearità, simmetria, giocatore fittizio e efficienza. Questi quattro assiomi si traducono in quattro concetti chiave per la spiegazione delle previsioni:

- **Linearità**: Se una funzione di predizione è la somma di due funzioni di predizione, allora la sua spiegazione è la somma delle spiegazioni delle due funzioni individuali.
- **Simmetria**: La stessa contribuzione di due caratteristiche diverse deve avere la stessa spiegazione. Questo è un requisito importante, perché se il metodo di spiegazione assegna due spiegazioni diverse a due caratteristiche che danno la stessa contribuzione alla predizione, allora questo metodo di spiegazione non può essere considerato affidabile.
- **Giocatore fittizio**: Una caratteristica che non contribuisce alla predizione deve avere un valore di spiegazione pari a zero.
- **Efficienza**: Quello che non è spiegato dalla previsione media globale è completamente spiegato dalle caratteristiche, ovvero considerando che dalla Eq. (4.1), $f(x) - \phi_0 = \sum_{i=1}^{n} \phi_i$, allora la somma dei valori di Shapley per le caratteristiche è uguale alla differenza tra il valore di previsione e la previsione media globale. Questa proprietà, nel contesto della spiegazione delle previsioni, è anche nota come completezza

4.2.1 Calcolo numerico dello Shapley Value

In ambito di spiegazione delle previsioni, i valori di Shapley rappresentano l'unico metodo di attribuzione delle caratteristiche che rispetta gli assiomi della teoria dei giochi coalizionali. Il calcolo degli Shapley value è tuttavia complesso per due motivi principali: da una parte, è necessario calcolare l'aspettativa marginale di ogni giocatore, dall'altra, occorre effettuare la valutazione per ogni possibile coalizione.
Ci si focalizzi sul primo punto, ovvero il calcolo del contributo marginale di un giocatore i. Per poter calcolare gli Shapley value in modo pratico per ogni caratteristica, considerando un determinato modello e una specifica previsione generata dal modello, è necessario formulare ulteriori ipotesi. Infatti, per calcolare i valori di Shapley attraverso l'Eq. (4.3), bisogna calcolare $v(S) = \mathbb{E}[f(x)|x_S = x_S^*]$. In altre parole, per identificare il contributo marginale di un giocatore i, bisogna valutare il valore atteso di $f(x)$ condizionato alla variabile x_S che assume il valore x_S^*. Questo valore atteso può essere calcolato anche considerando il complemento S^C di S

$$v(S) = \mathbb{E}[f(x)|x_S = x_S^*] = \mathbb{E}[f(x_S, x_{S^C})|x_S = x_S^*] \tag{4.4}$$

$$= \int f(x_S, x_{S^C}) p(x_{S^C}|x_S = x_S^*) dx_{S^C} \tag{4.5}$$

dove $p(x_{S^C}|x_S = x_S^*)$ è la distribuzione condizionale di x_{S^C} dato $x_S = x_S^*$. La distribuzione condizionale si conosce solo in casi speciali. Nel caso di un modello di regressione lineare con pesi dati da w_i, per $i = 1, ..., n$ ovvero

$$f(x) = w_0 + \sum_{i=1}^{n} w_i x_i$$

dove $\phi_0(f) = w_0$ e assumendo le features indipendenti si ha

$$\begin{aligned} v(S) &= \int f(x_S, x_{S^C}) p(x_{S^C}|x_S = x_S^*) dx_{S^C} \\ &= \int \left(\sum_{i \in S} w_i x_i + \sum_{i \notin S} w_i x_i \right) p(x_{S^C}|x_S = x_S^*) dx_{S^C} \\ &= \sum_{i \in S} w_i \int x_i p(x_i) dx_i + \sum_{i \in S^C} w_i x_i^* \int p(x_{S^C}) dx_{S^C} \\ &= \sum_{i \in S} w_i \mathbb{E}[x_i] + \sum_{i \in S^C} w_i x_i^* \end{aligned}$$

Pertanto, $v(S \cup \{i\}) = v(S) + w_i(x_i^* - \mathbb{E}[x_i])$ e Eq.(4.3) diventa

$$\phi_i(f) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} w_i(x_i^* - \mathbb{E}[x_i]) = w_i(x_i^* - \mathbb{E}[x_i])$$

in quanto la somma dei pesi di Shapley è uguale a 1.

Questo esempio dimostra che in alcune situazioni è possibile calcolare gli Shapley value in modo esplicito, in quanto si può calcolare il vettore $v(S)$ dall'equazione (4.5). Tuttavia, in generale la distribuzione condizionale in questa equazione non è nota e sono necessarie ulteriori ipotesi per calcolare gli Shapley value.
Inoltre, per calcolare i valori di Shapley per ciascuna caratteristica, è necessario calcolare la differenza $v(S \cup \{i\}) - v(S)$ per tutte le possibili coalizioni (ovvero i possibili sottoinsiemi di caratteristiche) nell'equazione (4.3). Poiché il numero di caratteristiche può essere molto elevato, il numero di valutazioni richieste per calcolare i valori di Shapley cresce esponenzialmente, il che rende il calcolo proibitivo quando il numero di caratteristiche diventa molto grande.

4.3 TreeSHAP

TreeShap [23] è un metodo di spiegazione specifico per modelli basati su alberi, come ad esempio gli alberi decisionali, le random forest e le macchine di boosting del gradiente. A differenza di KernelShap, si basa sugli Shapley value, ma sfrutta la struttura dei modelli basati su alberi per calcolarli in modo efficiente. Per ogni caratteristica, TreeShap calcola gli Shapley value esatti attraverso una ricorsione nel modello ad albero, attribuendo contributi a ciascuna caratteristica man mano che scende nell'albero. Utilizza un approccio di programmazione dinamica per evitare calcoli ridondanti e ridurre la complessità computazionale.

4.3.1 Funzionamento

L'algoritmo TreeShap segue i seguenti passaggi:

1. Attraversa l'albero partendo dalla radice fino ai nodi foglia, tenendo traccia del percorso di decisione per l'istanza di interesse.
2. Assegna contributi a ciascuna caratteristica incontrata lungo il percorso, considerando il numero di possibili combinazioni delle caratteristiche e la probabilità di ciascuna combinazione.
3. Il processo è ripetuto per tutti gli alberi dell'insieme, se applicabile.
4. Calcola la media dei contributi di tutti gli alberi per ottenere gli Shapley value finali.

4.3.2 Vantaggi e Svantaggi

I vantaggi di TreeShap:

- Calcola gli Shapley value esatti senza la necessità di campionamenti o approssimazioni.
- È efficiente dal punto di vista computazionale grazie al suo approccio di programmazione dinamica.
- È appositamente progettato per modelli basati su alberi, i quali sono ampiamente utilizzati nella pratica.

Svantaggi:
TreeShap è limitato ai modelli basati su alberi e non può essere applicato ad altri tipi di modelli, come quelli di deep learning o le SVM.

4.4 KernelSHAP

Kernel SHAP è un metodo di spiegazione che funziona su diversi tipi di modelli ed è stato sviluppato combinando LIME e SHAP [6].
Il vettore x rappresenta l'input del modello f che deve essere spiegato, mentre il vettore binario x' rappresenta l'interpretazione dell'input x, ovvero quali feature sono presenti o assenti. Ad esempio, nel caso di immagini, l'interpretazione può essere rappresentata da un vettore binario che indica quali superpixel sono stati selezionati. G rappresenta la classe di modelli interpretabili e g rappresenta un modello di spiegazione all'interno di questa classe. L'algoritmo LIME restituisce una spiegazione che tiene conto della fedeltà al modello originale e della complessità del modello di spiegazione

$$\xi = \arg\min_{g \in \mathcal{G}} \mathcal{L}(f, g, \pi_{x'}) + \Omega(g) \tag{4.6}$$

Dove $\pi_x(z)$ rappresenta la misura di vicinanza tra un'istanza z e x, dove $z' \in \{0,1\}^m$ è un vettore che contiene una frazione degli elementi non nulli di x'. Queste istanze z' sono campionate casualmente e uniformemente attorno a x'. La funzione di perdita L, il kernel π_x e il termine di regolarizzazione Ω sono scelti in modo euristico. In generale, la spiegazione data dall'equazione precedente non è uguale agli Shapley value. LIME approssima i valori di Shapley se si assume che il modello di spiegazione g nell'espressione precedente sia lineare e la funzione di perdita $\mathcal{L}$, il kernel $\pi_{x'}$ e Ω sono dati da

$$\Omega(g) = 0 \tag{4.7}$$

$$\pi_{x'}(z') = \frac{m-1}{\binom{m}{|S|}|S|(m-|S|)} \tag{4.8}$$

$$\mathcal{L}(f, g, \pi_{x'}) = \sum_{z' \in Z} [f(h_x(z')) - g(z')]^2 \pi_{x'}(z') \tag{4.9}$$

dove h_x è la funzione che mappa il vettore binario degli input interpretabili nello spazio degli input originali e $f(h_x(z')) = v(S)$ è dato dall'Eq.(4.2), dove S indica l'insieme degli indici non nulli in z'.
È possibile riscrivere la loss in termini di coalizioni utilizzando l'uguaglianza fornita e considerando il fatto che questa perdita restituisce gli Shapley value come modello di spiegazione. In altre parole, è possibile esprimere la perdita in termini di gruppi di elementi che collaborano insieme, anziché in termini di singoli elementi isolati.

$$\mathcal{L} = \sum_{S \subseteq N} \pi_{x'}(z') \left(v(S) - \left(\phi_0 - \sum_{i \in S} \phi_i \right) \right)^2 \tag{4.10}$$

KernelSHAP utilizza un'ipotesi di indipendenza tra le caratteristiche di input per approssimare gli Shapley value, e quindi stima il valore di v tramite l'equazione (4.5). Inoltre, assume la linearità del modello, tale che

$$v(s) \approx \mathbb{E}[f(x_S, x_{S^C})|x_S = x_S^*] = f(x_S, \mathbb{E}[x_{S^C}]) = \frac{1}{K} \sum_{i=1}^{K} f(x_S^*, x_{S^C}^k) \tag{4.11}$$

dove ogni $x_{S^C}^k$, per $i = 1, ..., K$ indica un campione preso dai dati di addestramento.
La funzione kernel π viene definita in modo tale che, se $|z'|$ appartiene all'insieme $\{0, m\}$, allora $\pi_{x'}(z') = \infty$. Per evitare problemi numerici associati a questi casi, il valore di π viene impostato su una costante elevata. Ciò garantisce che $\phi_0 = v(\emptyset)$, ovvero lo Shapley value associato all'insieme vuoto, e che la somma degli Shapley value sia uguale a $f(x)$, la previsione del modello per l'input x in questione.
Il modello utilizza una loss quadratica, il che consente di calcolare gli Shapley value attraverso la soluzione di un problema di minimizzazione dei minimi quadrati. In pratica, viene utilizzata una matrice binaria Z per rappresentare le possibili combinazioni di presenza/assenza delle caratteristiche, dove la prima colonna è costituita da 1 e gli elementi nelle righe successive indicano la presenza o l'assenza della caratteristica corrispondente alla colonna. V viene utilizzato come vettore contenente gli Shapley value, mentre W è una matrice diagonale che include i pesi del kernel π. A questo punto, il problema può

essere espresso come un problema di minimizzazione dei minimi quadrati:

$$(v - Z\phi)^T W(v - Z\phi).$$

4.5 Campionamento di Shapley

Il calcolo dei valori di Shapley presenta una complessità computazionale elevata, che non può essere facilmente ridotta. Tuttavia, esiste un metodo di approssimazione degli Shapley value ancora all'interno del framework SHAP [20], noto come metodo di campionamento di Shapley [4]. L'idea alla base di questo metodo è quella di utilizzare la procedura di campionamento per estrarre un campione rappresentativo di una particolare popolazione quando non è possibile ottenere informazioni da ciascun membro della popolazione.
Per definire il metodo di campionamento di Shapley, si parte dalla formulazione degli Shapley value come valori probabilistici nella formula ((3.17)).
Questa formulazione può essere espressa anche in termini di tutte le possibili permutazioni dei giocatori. In particolare, per un insieme di N giocatori, $\pi(N)$ è l'insieme di tutte le possibili permutazioni dei giocatori.
Il numero di tutte le possibili permutazioni è $|\pi(N)| = n!$. Sia O un ordine in $\pi(N)$ tale che $O(k)$ sia il giocatore assegnato alla posizione k e sia $Pred^i(O)$ l'insieme dei predecessori del giocatore i nell'ordine O. Gli Shapley value possono quindi essere espressi come:

$$\phi_i(v) = \frac{1}{n!}(v(Pred^i(O) \cup \{i\}) - v(Pred^i(O))) \tag{4.12}$$

per $i = 1, ..., N$.
La funzione caratteristica v viene definita come la predizione del modello basata sulle caratteristiche presenti in $Pred^i(O)$, ovvero $v(Pred^i(O)) = f(x_{Pred^i(O)})$. Per approssimare i valori di Shapley si utilizza l'approssimazione di Shapley sampling, la quale considera $\pi(N)$ come popolazione per il processo di campionamento e, scegliendo una dimensione del campione m, si seleziona un campione M che rappresenta un insieme di combinazioni di lunghezza m (dentro l'insieme $\underbrace{P \times P \times \cdots \times P}_{m\text{-times}}$). I valori di Shapley vengono poi approssimati calcolando la media dei contributi marginali dei singoli elementi di M.

$$\phi_i(v) = \frac{1}{m} \sum_{O \in M} (v(Pred^i(O) \cup \{i\}) - v(Pred^i(O)))$$

dove ogni ordine $O \in \pi(N)$ è scelto con una probabilità di $\frac{1}{n!}$.
L'uso di questa approssimazione garantisce che il calcolo del valore di Shapley richieda un tempo polinomiale se v può essere calcolato in tempo polinomiale. Inoltre si può dimostrare che questo approssimatore degli Shapley value è privo di bias ($\mathbb{E}[\phi_i] = \phi_i$, con varianza data da $Var(\phi_i) = \frac{\sigma^2}{m}$, dove $\sigma^2 = \sum_{O \in \pi(N)} \frac{1}{n!}(v(Pred^i(O) \cup \{i\}) - v(Pred^i(O)) - \phi_i)^2$. Pertanto, $\phi_i \approx \mathcal{N}(\phi_i; \frac{\sigma^2}{m}$. Questo approssimatore soddisfa gli assiomi si giocatore nullo ed efficienza.

4.6 LIME (Spiegazioni Locali Interpretabili Indipendenti dal Modello)

I modelli surrogati locali sono modelli interpretabili utilizzati per spiegare le singole previsioni dei modelli di machine learning "black box".
LIME (Local Interpretable Model-Agnostic Explanations) [16] è stato proposto come implementazione concreta di modelli surrogati locali. Questi modelli surrogati vengono addestrati per approssimare le previsioni del modello black box sottostante. A differenza di un modello surrogato globale, LIME si concentra sull'addestramento di modelli surrogati locali per spiegare le singole previsioni.

LIME è un metodo che cerca di spiegare il comportamento di un modello di machine learning "black box" senza accedere ai dati di addestramento. Si fa ciò interrogando il modello, introducendo variazioni nei dati di input e creando un modello interpretabile locale basato su queste risposte. Questo modello interpretabile aiuta a comprendere quali feature influenzano le previsioni del modello black box per un determinato input. Esso crea un nuovo insieme di dati composto da campioni perturbati e le relative previsioni del modello black box. Successivamente, vi addestra un modello interpretabile, attribuendo un peso in base alla vicinanza tra le istanze campionate e l'istanza di interesse. Il modello interpretabile può essere scelto tra diversi modelli interpretabili, come ad esempio una random forest o un SVM. L'obiettivo è che il modello appreso sia una buona approssimazione delle previsioni del modello di machine learning a livello locale, senza la necessità di essere una buona approssimazione globale. Questo tipo di accuratezza è comunemente definita fedeltà locale.
Dal punto di vista matematico, i modelli surrogati locali con vincoli di interpretabilità possono essere descritti come segue:

$$spiegazione(x) = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

Il modello di spiegazione per l'istanza x è rappresentato dal modello g (come ad esempio un modello di regressione lineare), che viene selezionato in modo da minimizzare la loss L (come ad esempio l'errore quadratico medio). La loss misura quanto vicina sia la spiegazione rispetto alla previsione del modello originale f (come ad esempio un modello xgboost), mantenendo al contempo bassa la complessità del modello $\Omega(g)$ (preferendo ad esempio un numero ridotto di caratteristiche). G rappresenta la famiglia di possibili spiegazioni, come ad esempio tutti i possibili modelli di regressione lineare. La misura di prossimità π_x definisce l'ampiezza del vicinato intorno all'istanza x che viene considerato per la spiegazione. Nella pratica, LIME si concentra principalmente sull'ottimizzazione della los. Spetta all'utente determinare la complessità, ad esempio selezionando il numero massimo di caratteristiche che il modello di regressione lineare può utilizzare.
Ecco i passi per addestrare i modelli surrogati locali:

- Selezionare l'istanza di interesse per cui si desidera ottenere una spiegazione della sua previsione del modello black box.
- Perturbare il dataset e ottenere le previsioni del modello black box per questi nuovi punti.

- Assegnare un peso ai nuovi campioni in base alla loro vicinanza all'istanza di interesse.
- Addestrare un modello interpretabile ponderato utilizzando il dataset con le variazioni.
- Spiegare la previsione interpretando il modello locale.

Nell'implementazione attuale in R e Python, ad esempio, è possibile scegliere la regressione lineare come modello surrogato interpretabile. Prima di iniziare, è necessario selezionare il valore di K, che rappresenta il numero massimo di caratteristiche che si desidera includere nel modello interpretabile. Un valore K più basso rende il modello più facile da interpretare, mentre un valore K più alto potrebbe produrre modelli con una migliore precisione. Esistono diversi metodi per addestrare modelli con esattamente K caratteristiche. Ad esempio, si può utilizzare il metodo Lasso, che, con un parametro di regolarizzazione λ elevato, porterà a un modello senza alcuna caratteristica. Riducendo gradualmente il valore di λ durante il processo di addestramento dei modelli Lasso, le caratteristiche otterranno stime dei pesi diverse da zero. Quando il modello contiene K caratteristiche, si raggiunge il numero desiderato. Altre strategie includono la selezione delle caratteristiche in avanti o all'indietro, in cui si parte con un modello completo (contenente tutte le caratteristiche) o con un modello con solo l'intercetta e si valuta quale caratteristica contribuisce maggiormente all'incremento delle prestazioni, aggiungendola o rimuovendola, fino a ottenere un modello con K caratteristiche.
Per quanto riguarda la generazione delle variazioni dei dati, dipende dal tipo di dati considerato: testo, immagini o dati tabulari. Nel caso di testo e immagini, è possibile attivare o disattivare parole singole o super-pixel. Per i dati tabulari, LIME genera nuovi campioni perturbando individualmente ciascuna caratteristica, utilizzando una distribuzione normale con media e deviazione standard derivate dalla caratteristica stessa.

4.7 Applicazioni in contesti reali

Negli ultimi dieci anni i metodi di spiegazione basati sui valori di Shapley hanno attirato molta attenzione grazie all'interesse per le reti neurali. Questi metodi hanno il vantaggio di essere basati sulla teoria matematica dei giochi coalizionali. Tuttavia, il loro utilizzo in applicazioni reali ha generato dibattiti tra i ricercatori.
Uno dei punti di discussione riguarda il tipo di assunzioni fatte per approssimare gli Shapley value, in particolare per l'uso di una attesa condizionata o interventistica per approssimare il contributo marginale. Un secondo punto dibattuto riguarda il fatto che la spiegazione fornita dagli Shapley value non corrisponde a quanto ci si aspetterebbe da un utente umano.

4.7.1 Attesa condizionale e attesa interventiva

Il primo punto da considerare riguarda il calcolo approssimato degli Shapley value per ogni feature, il quale richiede la valutazione della contribuzione marginale della feature

a tutti i possibili sottoinsiemi di feature. Questo calcolo è troppo costoso computazionalmente, quindi è necessario fare delle approssimazioni per calcolare gli Shapley value. Alcuni modelli definiscono le contribuzioni marginali in termini di aspettativa condizionata della previsione del modello, assumendo che solo le feature x_S siano note, ovvero $\mathbb{E}_D[f(x)|X_S = x_S)]$.
Alcuni modelli utilizzano l'attesa condizionata della previsione del modello per definire le contribuzioni marginali ai fini del calcolo degli Shapley values, mentre altri introducono condizioni sui valori delle feature e stimano l'aspettativa marginale su una distribuzione interventiva. Ad esempio, KernelSHAP assume l'indipendenza delle feature e stima l'attesa condizionata mediante il calcolo di $\mathbb{E}_D[f(x_S, X_{S^C})]$, dove D è la distribuzione marginale congiunta delle X_{S^C}.
I metodi di spiegazione che stimano l'aspettativa condizionata per calcolare gli Shapley value sono chiamati "condizionali", mentre i metodi che si basano sulla stima dell'aspettativa su distribuzioni interventistiche sono definiti "interventistici". Ad esempio, KernelSHAP e Shapley Samplings sono interventistici, mentre TreeSHAP, che è un metodo SHAP adattato ai modelli basati su alberi è condizionale.
La discussione sul quale dei due metodi, interventistico o condizionale, sia migliore si basa sulla seguente considerazione: se per una feature x_i si ha $f(x) = f(x')$ dove $xj = x'_j$ per tutti $j \neq i$, che significa che cambiare solo la feature x_i non influisce sulla predizione, dovrebbe x_i essere considerata un giocatore nel gioco di spiegazione? Potrebbe sembrare opportuno rimuovere questa feature dal set di giocatori, ma se x_i è un proxy statistico per un'altra variabile x_k, potrebbe avere un effetto indiretto sulla predizione e la sua rimozione potrebbe avere un impatto anche sull'esplicazione. I metodi condizionali attribuiscono un punteggio diverso da zero a questo tipo di variabili, mentre quelli interventistici gli attribuiscono un punteggio pari a zero.
I metodi condizionali richiedono la conoscenza delle distribuzioni di tutte le combinazioni possibili delle features (2^n), il che comporta una notevole modellizzazione. Inoltre, per insiemi di feature molto grandi, l'aspettativa condizionale viene calcolata solo su un sottoinsieme di feature significativo per ridurre i costi computazionali. Tuttavia, la scelta di quali feature sono ridondanti e possono essere scartate per il calcolo non è scontata. Inoltre, non è ovvio se le feature statisticamente correlate debbano essere considerate giocatori separati o meno.
I metodi interventivi, invece, si basano sulla valutazione del modello su campioni che si trovano al di fuori delle distribuzioni delle feature, perché alcuni valori sono sostituiti nell'insieme delle feature complementari. Pertanto, il modello da spiegare potrebbe estrapolare alcune informazioni da queste feature non viste. Il metodo di spiegazione potrebbe quindi fornire informazioni indesiderate su questa estrapolazione, che non riflettono la vera spiegazione della previsione del modello.

4.7.2 Spiegazione adattata all'essere umano

In questa sezione, si affronta il punto di vista centrato sull'uomo riguardo all'argomento delle spiegazioni. Le spiegazioni basate sugli Shapley value sono state criticate anche perché le spiegazioni fornite non sembrano intuitive dal punto di vista umano. Le critiche si basano su tre punti principali:

1. le spiegazioni umane dei fenomeni si basano su affermazioni contrastive, il che significa che nel caso degli Shapley value, bisogna rispondere alla domanda su perché si è ottenuto $f(x)$ invece di $\mathbb{E}[f(x)]$. Tuttavia, a seconda del contesto, i valori attesi potrebbero non essere qualcosa che un utente si aspetterebbe di ottenere;

2. la spiegazione dovrebbe fornire all'utente un modo per raggiungere un risultato desiderato, ma gli Shapley value non forniscono un modo per modificare il valore di una funzione in modo che la previsione cambi di conseguenza;

3. la spiegazione dovrebbe aiutare l'utente a valutare la decisione presa dal modello, ma la conoscenza degli Shapley value da sola non sembra essere sufficiente per giustificarla.

Diversi studi hanno applicato con successo i metodi di spiegazione basati su Shapley values per valutare l'importanza delle caratteristiche e spiegare le previsioni dei modelli di deep learning. Inoltre in "n Proceedings of Machine Learning and Systems 2020" di (Kumar, Suresh Venkatasubramanian, Carlos Scheidegger, and Sorelle Friedler) è stata presentata un'analisi qualitativa dell'effetto di SHAP sul processo decisionale durante l'elaborazione degli allarmi, dove gli autori sostengono che i grandi valori SHAP focalizzano l'attenzione su alcune caratteristiche che altrimenti verrebbero ignorate. In "Interpretation of machine learning models ´ using shapley values: application to compound potency and multi-target activity predictions." di (R. Rodrıguez-Perez and J. Bajorath) è stata effettuata un'analisi di modelli DNN per la generazione di profili di attività multi-target per la predizione dell'attività e della potenza dei composti nei farmaci utilizzando SHAP. I loro risultati mostrano la validità dell'utilizzo di KernelSHAP per spiegare le caratteristiche del modello responsabili delle previsioni individuali. In "In Suzuki K. et al. Interpretability of Machine Intelligence in Medical Image Computing and Multimodal Learning for Clinical Decision Support" di (Young K., Booth G., Simpson B., Dutton R., and Shrapnel S), è stata presentata una ricerca sull'applicazione di KernelSHAP alla predizione della classificazione del melanoma sulle immagini di dermatoscopia generate da una rete neurale profonda, dove gli autori riportano che l'interpretazione umana della diagnosi del melanoma mostra una maggiore concordanza con le spiegazioni di SHAP rispetto alle spiegazioni fornite da LIME o DeepLIFT e che KernelSHAP è utile per scoprire le possibili fonti di bias nelle previsioni della rete.
Infine, l'interpretabilità di un modello di rete neurale profonda che prevede le serie temporali della qualità dell'aria utilizzando i metodi di spiegazione basati sugli Shapley value è stata riportata in "Shapley additive explanations for no2 ´ forecasting" di (Marıa Vega Garcıa and Jose L. Aznarte). Lo studio ha evidenziato che i metodi SHAP possono aiutare a comprendere meglio quali fattori hanno un impatto maggiore sulla qualità dell'aria e sul comportamento del modello nella previsione delle serie temporali dell'inquinamento.

Questi studi indicano che i metodi di spiegazione basati sui valori di Shapley possono essere utili per ottenere una comprensione di come funziona una rete neurale profonda e scoprire informazioni importanti sulle caratteristiche che hanno maggiormente influenzato l'output.

4.8 Altri Metodi di spiegazione

4.8.1 DeepSHAP

DeepLIFT è un metodo di spiegazione specifico del modello che utilizza perturbazioni e si avvale della struttura della rete per confrontare l'attivazione di ogni neurone con un riferimento chiamato "baseline" [19]. Attraverso questo confronto, vengono assegnati punteggi alle diverse caratteristiche in base alla loro differenza.
Combinando DeepLIFT e SHAP, è stato creato un nuovo metodo chiamato DeepSHAP, il quale stima gli Shapley value assumendo che le caratteristiche di input siano indipendenti e che il modello da spiegare sia lineare. Per calcolare le attribuzioni di ogni caratteristica, DeepLIFT considera la differenza tra l'uscita del neurone target $t(y)$ e quella ottenuta utilizzando un input di riferimento $t(y_0)$, assegnando poi valori di contributo $C_{\Delta_{y_i},\Delta_t}$ alla differenza in input Δ_{y_i} in modo da evidenziare l'importanza delle diverse caratteristiche

$$\sum_{i=1}^{K} C_{\Delta_{y_i},\Delta_t} = \Delta_t$$

Gli y_i, per $0 \leq i \leq K$ sono i neuroini di input usati per calcolare t. La "reference input" di solito è un vettore in cui alcune caratteristiche di input sono assenti (ad esempio, per un'immagine, il riferimento può essere un'immagine nera). I valori di contributo vengono calcolati definendo i moltiplicatori

$$m_{\Delta_y\Delta_t} = \frac{C_{\Delta_y\Delta_t}}{\Delta_y}$$

Successivamente, i valori dei moltiplicatori vengono propagati attraverso la rete utilizzando regole appropriate. Per fare ciò, viene utilizzata la regola della catena, che è simile alla regola della catena per le derivate e viene combinata con la regola del ridimensionamento e la regola della rivelazione-cancellazione. In questo modo, i valori di contributo delle diverse caratteristiche vengono calcolati e propagati attraverso la rete in maniera efficiente.
La regola del ridimensionamento è stata sviluppata per le trasformazioni non lineari con un unico input, come la funzione ReLU. Essa prevede che i moltiplicatori siano calcolati come il rapporto tra la differenza tra l'output della rete in corrispondenza dell'input di interesse e quello di riferimento e la differenza tra l'input di interesse e quello di riferimento ($m_{\Delta_y\Delta_t} = \frac{\Delta_t}{\Delta_x}$). In particolare, i moltiplicatori convergono alla derivata dell'output rispetto all'input di interesse valutata sull'input di riferimento. Questo metodo prende il nome di **Deeplift-Rescale**.
La regola della rivelazione-cancellazione è stata definita per le trasformazioni non lineari con un unico input e considera separatamente le contribuzioni positive e negative. Questa regola calcola la contribuzione dei termini positivi in assenza di termini negativi e la

contribuzione dei termini negativi in assenza di termini positivi. Questo metodo prende il nome di **Deeplift-RevealCancel**.
La regola Reveal-Cancel viene ri-definita per le trasformazioni non lineari con un singolo input, ma in questo caso le contribuzioni positive e negative sono considerate separatamente. In particolare, la regola valuta la contribuzione dei termini positivi in assenza di termini negativi, e quella dei termini negativi in assenza di termini positivi. Questo metodo prende il nome di Deeplift-RevealCancel. Nel contesto di modelli lineari e feature indipendenti, DeepSHAP definisce i moltiplicatori utilizzando i valori di Shapley come punto di riferimento

$$m_{x_i f} = \frac{\phi_i}{x_i - \mathbb{E}[x_i]} \tag{4.13}$$

Successivamente, l'algoritmo viene applicato alla rete neurale fornita utilizzando le regole di propagazione descritte in precedenza per la retropropagazione dei gradienti. In questo modo si ottiene un'approssimazione dei valori di Shapley per ogni feature di input

$$\phi_i \approx m_{x_i f}(x_i - \mathbb{E}[x_i])$$

come per il modello lineare.

4.8.2 DASP (Propagazione di Shapley Approssimata Profonda)

Deep approximate Shapley propagation (DASP) è un metodo di spiegazione che usa perturbazioni per approssimare gli Shapley value di una rete neurale profonda in un tempo relativamente breve [3]. Questo metodo di spiegazione è specifico per le reti neurali profonde, in quanto utilizza la struttura della rete per calcolare le aspettative marginali nella formula dello Shapley value.
Il metodo è basato sul fatto che per ogni caratteristica, lo Shapley value viene calcolato facendo la media dei valori del contributo marginale di tale caratteristica a tutte le possibili coalizioni di caratteristiche. Invece di calcolare il contributo a ciascuna coalizione, DASP calcola il contributo a una coalizione casuale. Per ogni caratteristica x_i e per ogni possibile dimensione di coalizione $0 \leq k \leq n-1$, DASP calcola il contributo atteso di quella caratteristica rispetto alla distribuzione delle coalizioni che non la contengono. Infine approssima lo Shapley value per ogni caratteristica facendo la media di questi contributi:

$$\mathbb{E}[R_i] = \frac{1}{n} \sum_{k=0}^{n-1} \mathbb{E}_k[R_{i,k}] \tag{4.14}$$

dove $\mathbb{E}_k$ è il valore atteso calcolato su insiemi di distribuzione di dimensione k e R_i rappresenta il contributo della caratteristica x_i ad una coalizione randomica di dimensione k:

$$R_{i,k} = f(S \cup \{i\}) - f(S) \;\; \text{con } |S| = k \tag{4.15}$$

la funzione f rappresenta il risultato della rete neurale quando conosce solo un sottoinsieme delle caratteristiche, ovvero solo x_S e le altre sono impostate a 0. Questo concetto di "base" implicitamente indica l'assenza di informazioni sulle caratteristiche impostate a 0 ed è presente in molti altri metodi di spiegazione, come DeepLIFT.

L'obiettivo è quindi di calcolare i valori attesi nell'Equazione (4.14) per ogni $k \in \{0, ..., n-1\}$. Per farlo, si utilizza un approccio di perturbazione che trasmette i valori da un livello all'altro della rete neurale, utilizzando una Lightweight probabilistic network. Questa architettura permette di propagare l'incertezza in una rete neurale, assumendo che l'input sia distribuito secondo una distribuzione gaussiana univariata e che i parametri del modello siano deterministici.
Si denoti con x_S un vettore di caratteristiche con valori $x_j = 0$ per $j \notin S$, dove $|S| = k$, per ogni $k \in \{0, ..., n-1\}$. Si suppone che questi vettori siano estratti da una distribuzione sottostante della variabile casuale X_k. Ogni unità nascosta Z_j nel primo strato di una rete neurale profonda è una somma pesata delle caratteristiche di input, ovvero $z_j = \sum_{i=1}^{n} w_{ij} x_i$. Poiché X_k è una variabile casuale, z_j è anche una variabile casuale con distribuzione

$$Z_j \approx \mathcal{N}\left(k\mu_j, k\sigma_j^2 \frac{n-k}{n-1}\right)$$

dove $\mu_j = \frac{1}{n}\sum_{i=1}^{n} w_{ij} x_i$ e $\sigma_j^2 = \frac{1}{n}\sum_{i=1}^{n}(w_{ij} x_i)^2 - \mu_j^2$. La distribuzione Z come input probabilistico viene propagata attraverso la rete utilizzando la lightweight probabilistic network. La lightweight probabilistic network converte ogni strato i di una data rete con input $X^{(i-1)}$ in uno strato di una rete probabilistica con input pari a una distribuzione gaussiana con media $\mu_x^{(i)} = \mathbb{E}_{x^{(i-1)}}[f^{(i)}(x^{(i-1)})]$ e varianza $\sigma^{2(i)}_X = V_{x^{(i-1)}}[f(i)(x^{(i-1)})]$.
L'intera rete di input viene trasformata in una rete probabilistica, dove l'output è rappresentato dai parametri della distribuzione all'ultimo livello. Successivamente, gli Shapley value vengono approssimati calcolando il contributo di ogni feature all'interno di una coalizione casuale di dimensione k. L'algoritmo richiede un costo computazionale di $O(n^2)$ per valutare la rete e approssimare gli Shapley value. Per ridurre il costo, si introduce una seconda approssimazione che considera solo un numero ridotto di dimensioni di coalizione, portando così il costo computazionale a $O(Kn)$ valutazioni della rete.

4.8.3 ANCHORS

Il metodo Anchors fornisce spiegazioni per le singole previsioni di qualsiasi modello di classificazione black box identificando una regola decisionale che "ancora" in modo sufficiente la previsione [15]. Una regola ancorata a una previsione rimane valida anche quando altre caratteristiche cambiano. Per raggiungere ciò, Anchors utilizza tecniche di apprendimento per rinforzo insieme a un algoritmo di ricerca del grafo per ridurre al minimo il numero di chiamate al modello e garantire una rapida esecuzione, evitando gli ottimi locali. Nel 2018, Ribeiro, Singh e Guestrin hanno proposto questo algoritmo, gli stessi ricercatori che hanno introdotto l'algoritmo LIME.
A differenza di LIME, che utilizza modelli surrogati, Anchors genera spiegazioni locali per le previsioni dei modelli black box sotto forma di regole IF-THEN facili da comprendere, chiamate anchors. Queste regole sono specifiche e possono essere riutilizzate in quanto indicano esattamente a quali altre istanze, anche non viste in precedenza, si applicano. La ricerca di anchors si basa su una strategia di esplorazione o di problema dei banditi multi-braccia, un concetto derivante dall'apprendimento per rinforzo. Per fare ciò, vengono creati e valutati vicini o perturbazioni per ogni istanza che si desidera spiegare. Questo approccio consente di ignorare la struttura interna e i parametri del modello black box,

mantenendoli non osservati e inalterati. Di conseguenza, l'algoritmo è indipendente dal modello, il che significa che può essere applicato a qualsiasi tipo di modello.
Nel loro studio, gli autori confrontano i due algoritmi e visualizzano come LIME e Anchors esplorano il vicinato di un'istanza in modo diverso per ottenere i risultati. Ad esempio, la figura illustra come LIME e Anchors spieghino in modo locale un classificatore binario complesso (che prevede solo - o +) utilizzando due istanze di esempio. I risultati di LIME non indicano la loro fedeltà, poiché LIME apprende solo un confine decisionale lineare che approssima il modello nel contesto di uno spazio di perturbazione specifico. Invece, Anchors costruisce spiegazioni che si adattano al comportamento del modello e presenta chiaramente i confini delle regole. Pertanto, le spiegazioni di Anchors sono intrinsecamente fedeli e indicano specificamente per quali istanze sono valide. Questa caratteristica rende Anchors particolarmente intuitivo e facile da comprendere.
I risultati o le spiegazioni dell'algoritmo pertanto vengono visualizzati sotto forma di regole chiamate anchors. Si prenda ad esempio un modello black box bivariato che prevede se un'applicazione mobile sarà scaricata o meno da un utente. Supponiamo di avere un'applicazione specifica e vogliamo capire perché il modello prevede che venga scaricata da un utente specifico. L'algoritmo Anchors fornirebbe una spiegazione del risultato nella forma di un anchor, come mostrato di seguito.

Feature	Valore
Età	23
Sesso	Maschio
Nazionalità	Italiana
Professione	Studente
Altri attributi	...
Applicazione	Sì

E la spiegazione corrispondente è:
IF *Nazionalità = Italiana* AND *Professione = Studente* THEN
PREDICT *Applicazione = Sì* WITH PRECISION 85% AND COVERAGE 20%
L'esempio dimostra come gli anchors possano fornire conoscenze essenziali sulla previsione di un modello e sul suo ragionamento sottostante. Il risultato indica quali attributi sono stati considerati dal modello, come ad esempio la nazionalità italiana e la professione sudente. Gli esseri umani, essendo di fondamentale importanza per la correttezza, possono utilizzare questa regola per convalidare il comportamento del modello. Inoltre, l'anchor ci informa che si applica al 20% delle istanze nel contesto dello spazio di perturbazione. In quei casi, l'esplicazione è accurata al 85%, il che significa che i predicati mostrati sono responsabili quasi esclusivamente dell'outcome previsto. Un anchor A è definito formalmente come segue:

$$\mathbb{E}_{\mathcal{D}_x(z|A)}[\mathbb{1}_{\hat{f}(x)=\hat{f}(x)}] \geq \tau,\ A(x) = 1$$

dove:

- x rappresenta l'istanza che sta venendo spiegata, ad esempio una riga in un insieme di dati tabulare.

- $D_x(\cdot|A)$ indica la distribuzione dei vicini di x che corrispondono ad A.
- f indica il modello di classificazione che deve essere spiegato, ad esempio un modello di rete neurale artificiale. Può essere interrogato per prevedere una label per x e le sue perturbazioni.
- $0 < \tau \leq 1$ specifica una soglia di precisione. Solo le regole che raggiungono una fedeltà locale di almeno τ vengono considerate come risultato valido.
- A è un insieme di predicati, ovvero la regola o l'anchor risultante, tale per cui $A(x)$ è uguale a 1 quando tutti i predicati delle caratteristiche definiti da A corrispondono ai valori delle caratteristiche di x.

La descrizione formale può essere riassunta come segue: Data un'istanza x da spiegare, si cerca di trovare una regola o un anchor A che si applichi a x e che preveda la stessa classe per almeno una frazione τ dei vicini di x in cui si applica lo stesso A. La precisione di una regola viene determinata valutando i vicini o le perturbazioni (seguendo $D_x(z|A)$) utilizzando il modello di machine learning fornito (indicato dalla funzione indicatrice $\mathbb{1}_{\hat{f}(x)=\hat{f}(z)}$).

4.8.4 Come trovare le regole Anchors

Nonostante il concetto di Anchors possa sembrare semplice, non è possibile creare delle regole specifiche. Questo richiederebbe di valutare l'espressione $1_{\hat{f}(x)=\hat{f}(z)}$ per ogni $z \in \mathcal{D}_x(\cdot|A)$, il che non è possibile in spazi di input continui o di grandi dimensioni. Di conseguenza, gli autori propongono di introdurre il parametro δ, con $0 \leq \delta \leq 1$, per creare una definizione probabilistica. In questo modo, vengono estratti campioni fino a quando non si raggiunge una fiducia statistica sulla loro precisione. La definizione probabilistica si presenta come segue:

$$\mathbb{P}(prec(A) \geq \tau) \geq 1 - \delta \text{ con } prec(A) = \mathbb{E}_{\mathcal{D}_x(z|A)}[1_{\hat{f}(x)=\hat{f}(z)}]$$

Le definizioni precedenti vengono messe insieme ed estese attraverso il concetto di copertura. La ragione dietro ciò è trovare regole che si applichino ad una ampio spettro dello spazio di input del modello. Formalmente, la copertura viene definita come la probabilità di un Anchor di essere applicato ai suoi vicini, ovvero allo spazio di perturbazione:

$$cov(A) = \mathbb{E}_{\mathcal{D}(z)}[A(z)]$$

Considerare questo elemento porta alla definizione finale dell'anchor, che tiene conto della massimizzazione della copertura:

$$\max_{A \text{ s.t. } \mathbb{P}(prec(A) \geq \tau) \geq 1-\delta} cov(A)$$

Pertanto, l'obiettivo del procedimento è trovare una regola che abbia la copertura più elevata tra tutte le regole ammissibili. Queste regole sono più significative poiché descrivono più ampiamente il modello. È da notare che le regole con un maggior numero di

predicati tendono ad avere una precisione superiore rispetto a quelle con meno predicati. In particolare, una regola che include tutte le caratteristiche di x riduce il vicinato valutato a istanze identiche, portando il modello a classificare tutti i vicini allo stesso modo e ottenendo una precisione del 100%. Allo stesso tempo, una regola che include molte caratteristiche risulta eccessivamente specifica e applicabile solo a poche istanze. Pertanto, c'è un bilanciamento tra precisione e copertura.

Componenti principali per la spiegazione

- Generazione dei candidati: Vengono generati nuovi candidati di spiegazione. Nella prima fase, viene creato un candidato per ogni caratteristica di x, fissando il valore delle possibili perturbazioni. Nei round successivi, i migliori candidati del round precedente vengono ampliati aggiungendo un predicato di caratteristica che non è ancora incluso.
- Identificazione miglior candidato: Nella fase di confronto delle regole candidate per spiegare x, vengono create perturbazioni che rispettano la regola attuale e valutate chiamando il modello. Per ridurre il carico computazionale, viene utilizzato un Multi-Armed-Bandit (MAB) di esplorazione pura. Ogni regola candidata viene considerata come un "braccio" che può essere attivato, fornendo informazioni sulla precisione della regola. La precisione rappresenta quanto bene la regola descrive l'istanza da spiegare.
- validazione della precisione dei candidati: Viene aumentato il numero di campioni nel caso in cui non vi sia ancora una sicurezza statistica che il candidato superi la soglia τ.
- Ricerca beam modificata: I componenti vengono combinati in una ricerca Beam (algoritmo di ricerca grafica basato sull'algoritmo di ricerca ad albero in ampiezza). Essa passa i migliori B candidati di ogni round al round successivo (con B larghezza del Beam). Esse vengono poi utilizzate per generare nuove regole. La ricerca Beam continua per un massimo di $featureCount(x)$ round, dato che ogni caratteristica può essere inclusa in una regola al massimo una volta. Ad ogni round i, vengono generati candidati con esattamente i predicati e vengono selezionati i migliori B tra di essi. Pertanto, impostando un valore elevato per B, l'algoritmo ha maggiori probabilità di evitare minimi locali. Tuttavia, ciò comporta un numero elevato di chiamate al modello e un aumento del carico computazionale.

4.8.5 Vantaggi e Svantaggi

Vantaggi

L'approccio basato sugli "anchors" offre diverse vantaggi. In primo luogo, l'output dell'algoritmo è più comprensibile, poiché le regole sono facili da interpretare, persino per persone non esperte nel campo. Inoltre, gli "anchors" sono suscettibili di essere suddivisi in sottoinsiemi e forniscono anche una misura di importanza attraverso il concetto di copertura. In secondo luogo, questo approccio funziona bene quando le previsioni del

modello sono non lineari o complesse nell'intorno di un'istanza. Poiché si basa su tecniche di apprendimento per rinforzo invece di adattare modelli surrogati, è meno probabile che il modello sia sottostimato. Oltre a ciò, l'algoritmo è indipendente dal modello utilizzato e quindi può essere applicato a qualsiasi tipo di modello. Inoltre, è altamente efficiente poiché può essere parallelizzato utilizzando MAB (Multi-Armed Bandit) che supportano il campionamento in batch, come ad esempio BatchSAR.

Svantaggi

L'algoritmo presenta alcune criticità, comuni a molti spiegatori basati sulla perturbazione. Non solo è necessario configurare accuratamente i parametri, come la larghezza del raggio o la soglia di precisione, per ottenere risultati significativi, ma è anche indispensabile progettare esplicitamente una funzione di perturbazione adatta a un determinato dominio o caso d'uso. Ad esempio, è necessario considerare come applicare la perturbazione ai dati tabulari e come adattare gli stessi concetti ai dati delle immagini. Fortunatamente, in alcuni domini, come in quello tabulare, possono essere utilizzati approcci predefiniti che semplificano l'implementazione iniziale delle spiegazioni.
Inoltre, molti scenari richiedono la discretizzazione dei dati, altrimenti i risultati ottenuti sarebbero troppo specifici, con una copertura limitata e non contribuirebbero a una comprensione approfondita del modello. Tuttavia, bisogna fare attenzione nell'utilizzare la discretizzazione, poiché può alterare i confini decisionali se impiegata in modo incauto, ottenendo quindi l'effetto contrario. Poiché non esiste una tecnica di discretizzazione universale, gli utenti devono considerare attentamente i dati prima di decidere come discretizzarli per evitare di ottenere risultati di scarsa qualità.
La costruzione degli "anchors" richiede molte chiamate al modello di machine learning, come tutti gli spiegatori basati sulla perturbazione. Nonostante l'algoritmo faccia uso delle Multi-Armed Bandits (MAB) per ridurre al minimo il numero di chiamate, il tempo di esecuzione dipende ancora molto dalle prestazioni del modello e può variare significativamente. Infine, il concetto di copertura non è definito in modo chiaro in alcuni domini. Ad esempio, non esiste una definizione evidente o universale per confrontare i "superpixel" di un'immagine con quelli presenti in altre immagini.

Capitolo 5

Esperimenti

Nel corso della ricerca condotta per questa tesi, è stata dedicata un'attenzione particolare alla valutazione delle performance degli algoritmi di spiegazione exactSHAP, treeSHAP, kernelSHAP e LIME, impiegando due modelli di machine learning ampiamente diffusi: XGBoost e Random Forest [1]. L'obiettivo principale di questo capitolo è quello di analizzare e confrontare le capacità di questi algoritmi nell'interpretare le previsioni di tali modelli, utilizzando due dataset differenti: il dataset Boston, disponibile all'indirizzo *"http://lib.stat.cmu.edu/datasets/boston"* e il dataset relativo ai dati del censimento degli adulti (Adult census data), reperibile attraverso la libreria *"shap.datasets.adult"*. La decisione di utilizzare dataset tabulari semplici con un numero limitato di feature anziché dataset visuali (come immagini) o testuali è stata presa considerando la comprensibilità e l'accessibilità da parte degli operatori umani. I dataset tabulari offrono una maggiore chiarezza e facilità di interpretazione, consentendo agli operatori umani di valutare in modo più efficace la relazione tra le feature e le previsioni dei modelli. In contrasto, i dataset di immagini o testo possono comportare un elevato numero di feature, complicando la comprensione del contributo di ciascuna feature alle previsioni. In aggiunta questo potrebbe compromettere la capacità di valutare la competizione tra le feature per le previsioni e, inoltre, aumentare significativamente la complessità computazionale rendendo difficile la conduzione degli esperimenti in modo efficiente. Pertanto, la scelta di dataset tabulari semplici è stata mirata a favorire una comprensione più chiara e una valutazione accurata dei metodi di spiegazione.
La scelta di condurre questi esperimenti su dataset diversificati è mirata a valutare l'adattabilità degli algoritmi di spiegazione a contesti eterogenei, sottolineando così la loro versatilità e robustezza in differenti contesti applicativi.
Il dataset Boston rappresenta un problema di regressione, in cui ci si propone di stimare il valore mediano delle case in quartieri residenziali di Boston sulla base di diverse caratteristiche, mentre il dataset del censimento degli adulti è un problema di classificazione, con l'obiettivo di determinare se un individuo guadagna più o meno di $50,000 all'anno a partire da vari attributi socio-demografici.
Questo capitolo presenta quindi un'analisi dettagliata delle performance di exactSHAP, treeSHAP, kernelSHAP e LIME in questi contesti diversificati, mettendo in evidenza la loro efficacia nell'offrire spiegazioni significative e affidabili per le predizioni dei modelli

XGBoost e Random Forest. I risultati di questi esperimenti costituiranno una componente fondamentale per la valutazione complessiva dei metodi di spiegazione considerati, contribuendo così alla comprensione della loro utilità e applicabilità in scenari reali di machine learning interpretativo.

5.1 Esperimento n°1 - Boston XGBoost

Nel primo esperimento di questa ricerca, ci si è concentrati sul dataset Boston, un set di dati ampiamente utilizzato nell'ambito della regressione, che mira a prevedere il valore mediano delle case in quartieri residenziali di Boston in base a varie caratteristiche socio-economiche. L'obiettivo principale di questo esperimento è stato valutare le performance degli algoritmi di spiegazione, senza applicare alcuna perturbazione ai dati, in un contesto di apprendimento supervisionato.
In particolare, è stato selezionato il modello di machine learning XGBoost come learner principale per la previsione del valore mediano delle abitazioni. XGBoost è noto per la sua capacità di gestire problemi di regressione e classificazione complessi, ed è ampiamente utilizzato in applicazioni del mondo reale. La scelta di questo learner mira a fornire un contesto significativo per valutare l'efficacia degli algoritmi di spiegazione nell'interpretazione di un modello di alto livello.
Gli algoritmi di spiegazione che sono stati testati in questo esperimento includono exactSHAP, treeSHAP, kernelSHAP e LIME. Essi sono stati applicati senza alcuna manipolazione o perturbazione dei dati di input, con l'obiettivo di valutare la loro capacità di fornire spiegazioni accurate e significative delle previsioni di XGBoost. Questo primo esperimento costituisce una base essenziale per comprendere come questi algoritmi si comportano in condizioni ideali, senza la presenza di rumore o disturbi nei dati.
I risultati di questo esperimento rappresenteranno un punto di riferimento cruciale per valutare le performance degli algoritmi di spiegazione e stabilire una linea di base per i confronti successivi. Inoltre, contribuiranno a fornire un'analisi preliminare dell'efficacia di tali algoritmi nell'interpretazione di modelli di machine learning in scenari reali, aiutando così a delineare l'approccio migliore per l'interpretazione dei risultati nei contesti applicativi futuri.

5.1.1 Dataset

Il Boston Housing Dataset deriva da informazioni raccolte dal Servizio Censimento degli Stati Uniti riguardo alle abitazioni nell'area di Boston, Massachusetts. Di seguito vengono descritte le colonne del dataset:

- **CRIM**: Questa colonna rappresenta il tasso di criminalità pro capite per città, fornendo informazioni sulla sicurezza e la criminalità in diversi quartieri.
- **ZN**: proporzione di terreni residenziali zonati per lotti superiori a 25.000 piedi quadrati. Indica l'entità delle proprietà residenziali di grandi dimensioni in ciascuna città.
- **INDUS**: proporzione di acri destinati a attività non commerciali per città. Può contribuire a valutare il carattere industriale delle diverse aree.
- **CHAS**: variabile dummy che assume valore 1 se un tratto di terra confina con il fiume Charles e 0 altrimenti. Fornisce informazioni sulla vicinanza a questa caratteristica naturale.

- **NOX**: concentrazione di ossidi di azoto, misurata in parti per 10 milioni. Misura la qualità dell'aria e i livelli di inquinamento nelle diverse aree.
- **RM**: numero medio di stanze per abitazione. Offre informazioni sulla dimensione e la disposizione delle abitazioni in vari quartieri.
- **AGE**: proporzione di unità abitative occupate dai proprietari costruite prima del 1940. Aiuta a valutare l'età e le condizioni del patrimonio abitativo.
- **DIS**: distanze ponderate verso cinque centri di impiego di Boston. Questo può indicare quanto le diverse aree siano accessibili a luoghi di lavoro.
- **RAD**: indice di accessibilità alle autostrade radiali. Fornisce informazioni sull'infrastruttura di trasporto in ciascuna città.
- **TAX**: tariffa di imposta sulla proprietà al valore pieno per $10.000. Riflette le aliquote fiscali immobiliari in diverse aree.
- **PTRATIO**: rapporto tra alunni e insegnanti per città. Offre informazioni sulla qualità dell'istruzione e sulle dimensioni delle classi nelle scuole locali.
- **B**: B è calcolata come 1000 volte la quantità $(Bk - 0.63)^2$, dove **Bk** rappresenta la proporzione della popolazione nera per città. Misura un aspetto socio-demografico della popolazione.
- **LSTAT**: percentuale dello status socio-economico più basso della popolazione. Fornisce informazioni sullo stato socio-economico dei residenti.
- **MEDV**: valore mediano delle abitazioni occupate dai proprietari, misurato in migliaia di dollari ($1000). È una variabile target chiave, che indica il valore immobiliare tipico in diversi quartieri.

Questo dataset è una risorsa preziosa per comprendere i vari fattori che influenzano i prezzi delle abitazioni e le caratteristiche dei quartieri, rendendolo una scelta popolare per compiti di regressione e modellizzazione predittiva nell'ambito dell'analisi dei dati e dell'apprendimento automatico.

5.1.2 Algoritmi di apprendimento utilizzati

XGBoost, che sta per *"eXtreme Gradient Boosting"* è un potente algoritmo di machine learning utilizzato sia per problemi di regressione che di classificazione. È noto per la sua efficienza e capacità di ottenere prestazioni eccezionali su una vasta gamma di dataset. Ecco come funziona XGBoost:

- **Boosting**: XGBoost è un algoritmo di tipo boosting. Il boosting è una tecnica di ensemble learning in cui vengono addestrati molti modelli deboli (solitamente alberi decisionali poco profondi) in sequenza, ognuno dei quali cerca di correggere gli errori del modello precedente. XGBoost addestra quindi una serie di alberi decisionali in modo incrementale, dove ciascun albero mira a migliorare la precisione complessiva del modello.

- **Alberi Decisionali**: Gli alberi decisionali sono gli "apprenditori deboli" di base utilizzati in XGBoost. Tuttavia, per migliorare l'efficacia dell'algoritmo, XGBoost adotta alcuni accorgimenti. Innanzitutto, i singoli alberi sono molto semplici, spesso alberi poco profondi chiamati "alberi di decisione regolari." Inoltre, XGBoost utilizza una versione migliorata dell'algoritmo di costruzione degli alberi per ottimizzare la suddivisione dei nodi.

- **Funzione di Perdita (Loss Function)**: XGBoost utilizza una funzione di perdita personalizzata, spesso chiamata "funzione di perdita XGBoost." Questa funzione misura quanto il modello si discosta dai valori target desiderati durante l'addestramento. L'obiettivo di XGBoost è minimizzare questa funzione di perdita.

- **Regularizzazione**: Per evitare il problema dell'overfitting (addestramento eccessivo), XGBoost incorpora tecniche di regolarizzazione nel processo di addestramento degli alberi. Queste tecniche includono la riduzione della complessità dell'albero, la limitazione della profondità massima e la penalizzazione dei pesi delle foglie.

- **Gradient Boosting**: La parte "Gradient" di XGBoost fa riferimento all'uso dei gradienti delle funzioni di perdita per determinare come adattare gli alberi successivi. In sostanza, XGBoost utilizza l'informazione derivata dalla funzione di perdita per guidare l'addestramento, concentrandosi sugli esempi che il modello sta gestendo in modo errato.

- **Riduzione di Overfitting**: XGBoost ha diversi parametri che possono essere regolati per evitare l'overfitting, come il tasso di apprendimento (learning rate), il numero massimo di iterazioni (numero di alberi), la profondità massima dell'albero e altri.

- **Feature Importance**: XGBoost calcola l'importanza delle feature durante l'addestramento, consentendo di identificare quali feature hanno un maggiore impatto sulle previsioni del modello.

- **Parallelismo**: XGBoost è progettato per il calcolo parallelo ed è noto per la sua efficienza computazionale, il che lo rende adatto all'addestramento su grandi dataset.

Complessivamente, XGBoost è un algoritmo di boosting altamente efficiente che combina molteplici modelli deboli per creare un modello complesso e ad alte prestazioni. Grazie alla sua versatilità e alla capacità di gestire una vasta gamma di problemi di machine learning, è diventato uno degli algoritmi più popolari nel campo dell'apprendimento automatico.

5.1.3 Risultati

Nel contesto di questo primo esperimento, ci si è concentrati sulla valutazione e il confronto degli algoritmi di spiegazione, tra cui exactSHAP, treeSHAP, kernelSHAP e LIME. L'obiettivo primario è stato comprendere l'efficacia di questi algoritmi nella spiegazione delle previsioni del modello, inizialmente attraverso il calcolo dell'Explanation Error, e in seguito analizzando la variazione delle performance in base a differenti metriche di selezione delle feature.

Per ottenere una panoramica iniziale delle prestazioni degli algoritmi, sono stati valutati i seguenti scenari:

- **"Keep Absolute"**: In questo scenario, le feature sono state selezionate in base al valore assoluto dell'influenza sull'output del modello. Le feature con una influenza assoluta più elevata sono state considerate più rilevanti per l'output del modello. Dopodiché è stato calcolato l'integrale della variazione dell'output del modello al variare del numero di feature con più influenza in valore assoluto coinvolte.
- **"Remove Absolute"**: In opposizione al caso precedente, le feature sono state rimosse in base al valore assoluto dell'influenza sull'output del modello. Le feature con una perdita assoluta più elevata sono state considerate meno rilevanti per l'output del modello.
- **"Keep Positive"**: In questo scenario, sono state mantenute solo le feature che contribuiscono positivamente all'output del modello. Le feature con un impatto positivo sull'output sono state considerate importanti per la previsione. In seguito è stato calcolato l'integrale della variazione dell'output del modello al variare del numero di feature con più influenza più positiva.
- **"Remove Positive"**: Al contrario, in questo caso, sono state rimosse le feature che avevano un impatto positivo sull'output del modello. Queste feature sono state considerate meno importanti o indesiderate per l'analisi delle metriche delle feature.
- **"Keep Negative"**: In questo scenario, vengono mantenute solo le feature che contribuiscono negativamente all'output del modello. Le feature con un impatto negativo sull'output sono considerate importanti per la previsione. Questo può significare che queste feature sono particolarmente significative per la previsione dei risultati.
- **"Remove Negative"**: Al contrario, in questo caso, vengono rimosse le feature che hanno un impatto negativo sull'output del modello. Queste feature sono considerate meno importanti o indesiderate per l'analisi delle metriche delle feature. Ciò potrebbe implicare che queste feature sono meno rilevanti per il modello o addirittura influenzano negativamente le previsioni.

In formule si ha:

$$M(x) = \int_0^1 f(x)dx - f(0)$$

- M indica la metrica (Keep/Remove Absolute/Positive/Negative).
- $f(x)$ rappresenta l'output del modello al variare della frazione di feature tenute o rimosse dalla maschera.
- x rappresenta la frazione di feature (tenute o rimosse) dalla mask che hanno un effetto maggiore (nel caso Keep) o minore (nel caso Remove) sull'output del modello.
- L'integrale $\int_0^1$ rappresenta l'area sotto la curva della variazione dell'output rispetto all'aumento delle feature (positive/negative/absolute) più influenti meno il valore dell'output nel caso in cui $x = 0$.

Inizialmente, questi scenari sono stati valutati attraverso la creazione di un grafico che ha visualizzato l'Explanation Error e la differenza nelle performance degli algoritmi in ciascun contesto di selezione delle feature.
Successivamente, per una comprensione ancora più dettagliata, sono stati generati grafici delle metriche con la variazione della frazione di feature mantenute o rimosse. Questo ha permesso di esplorare come le performance degli algoritmi si evolvono al variare del numero di feature coinvolte nell'analisi. Questo capitolo esaminerà approfonditamente i risultati di queste analisi, offrendo una visione chiara delle capacità degli algoritmi di spiegazione e delle dinamiche sottostanti che guidano la selezione e l'importanza delle feature nei contesti specifici.

Calcolo dell'Explanation Error

Il calcolo dell'Explanation Error (Errore di Spiegazione) è un processo che valuta quanto bene le attribuzioni SHAP spiegano le variazioni nelle previsioni di un modello. Qui si spiega brevemente il calcolo, includendo la notazione matematica:

1. **Inizializzazione dei Parametri:**
   - *masker*: Il mascheratore utilizzato per nascondere le feature.
   - *modello*: Il modello di machine learning.
   - *link*: La funzione di collegamento tra le spiegazioni SHAP e l'output del modello.
   - *batch_size*: La dimensione massima del batch per la predizione del modello.
   - *num_permutations*: Il numero di maschere casuali generate.
2. **Iterazione per le Spiegazioni (Attributi SHAP):** Per ogni attributo SHAP, l'importanza è rappresentata come $spiegazione_i$.
3. **Generazione di Maschere Casuali:** Vengono generate un numero specificato di maschere casuali per selezionare le feature da prendere in considerazione tra le feature con valore di spiegazione più basso, ciascuna identificata come $maschera\_casuale_j$.
4. **Predizione del Modello con Maschere:** Per ogni maschera casuale, viene calcolato il valore di output del modello quando la maschera viene applicata alle feature. Questo valore è rappresentato come $previsione\_con\_maschera_j$.
5. **Calcolo dell'Errore:** Si calcola l'errore tra la previsione del modello originale (senza maschere) e la previsione ottenuta con la maschera applicata. L'errore viene calcolato come la differenza tra i valori predetti:

$$(previsione\_originale - previsione\_con\_maschera_j)$$

6. **Misurazione dell'Errore:** L'errore tra le previsioni è misurato utilizzando una metrica di distanza. Nel caso fornito, l'errore è calcolato come il quadrato della differenza:

$$errore_j = (previsione\_originale - previsione\_con\_maschera_j)^2$$

7. **Calcolo dell'Errore di Spiegazione Complessivo:** L'errore di spiegazione complessivo viene ottenuto aggregando gli errori calcolati per tutte le maschere casuali. Questo può essere fatto calcolando la media degli errori quadrati:

$$errore\_complessivo = \frac{\sum_{j=1}^{n} errore_j}{n}$$

8. **Risultati:** Il risultato dell'errore di spiegazione per ciascuna spiegazione (attributo SHAP) viene restituito come parte di un oggetto BenchmarkResult, che contiene il nome della spiegazione e il valore dell'errore.

Questo processo misura quanto bene le attribuzioni SHAP spiegano le variazioni nelle previsioni del modello, confrontando le previsioni originali con quelle ottenute quando alcune feature vengono mascherate. L'Explanation Error è una misura dell'accuratezza delle attribuzioni SHAP nell'interpretazione del modello.

Risultati numerici

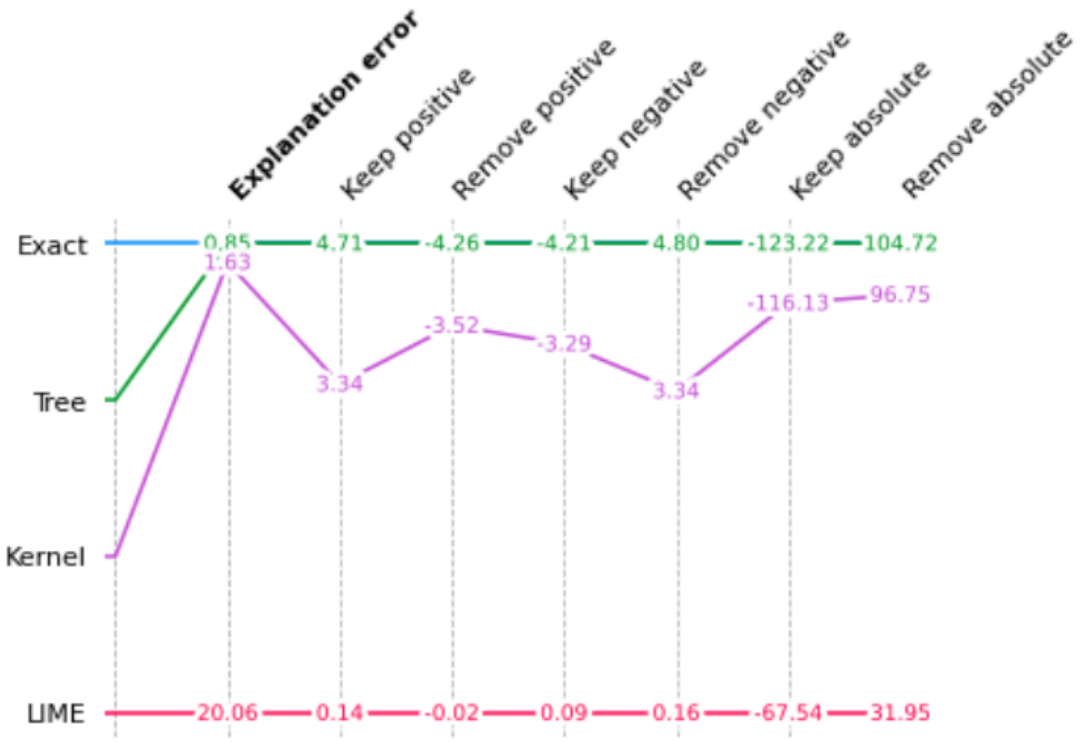


Figura 5.1: Explanation Error e integrale di variazione dell'output di ogni metrica

Nel grafico sopra (5.1), possiamo osservare le prestazioni di diversi metodi di calcolo dell'Explanation Error. È evidente che i risultati migliori sono stati ottenuti dai metodi di ExactSHAP e TreeSHAP, che si equivalgono nella precisione dei valori di Shapley in

quanto entrambi calcolano il valore esatto degli Shapley Value ma in maniera differente
Al secondo posto in termini di precisione c'è il metodo kernelSHAP, il quale utilizza un calcolo basato su campionamento, quindi, seppur meno preciso rispetto agli algoritmi esatti, fornisce comunque risultati affidabili. In contrasto, il metodo LIME si distingue per essere significativamente meno efficiente in ogni metrica presa in considerazione, questo potrebbe essere dovuto alla sua natura di modello di spiegazione basato su perturbazioni su singoli dati indipendenti dal modello di learning utilizzato.
Va notato che i valori nella colonna "Explanation Error" rappresentano direttamente l'errore di spiegazione calcolato per ciascun metodo, mentre le altre colonne mostrano l'integrale della variazione della previsione del modello in funzione delle frazioni di feature mantenute o rimosse. Questo grafico fornisce una chiara indicazione delle prestazioni relative dei diversi metodi nell'ambito del calcolo degli Shapley values e dell'Explanation Error.

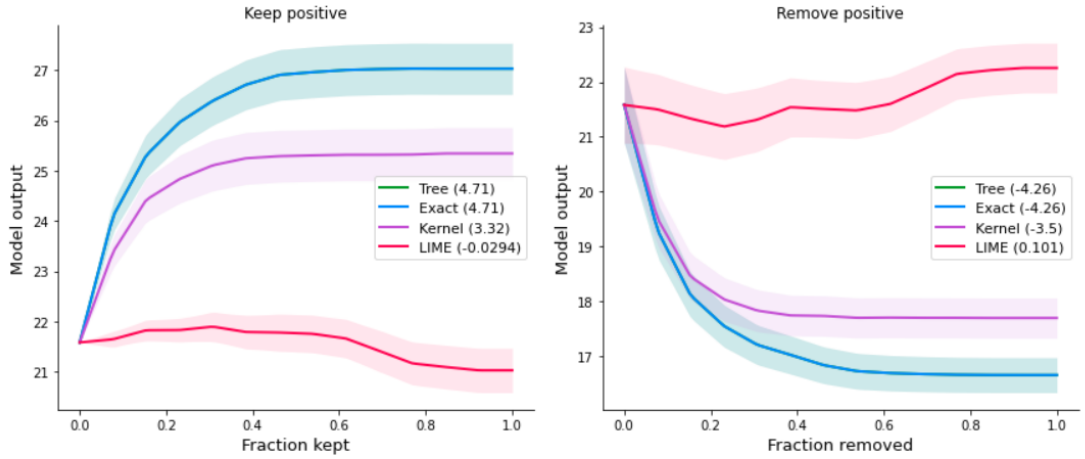


Figura 5.2: Variazione dell'output del modello lungo le metriche "Keep Positive" e "Remove Positive" al variare della frazione tenuta o rimossa

Nel grafico (5.2), esaminiamo l'influenza delle feature sull'output del modello. Due scenari sono confrontati: nel primo grafico, le feature più 'positivamente influenti' sono identificate e mantenute, mentre nel secondo grafico, queste stesse feature vengono rimosse. La variazione media nelle previsioni del modello è calcolata in entrambi i casi.
I metodi di spiegazione più efficaci, evidenziati da curve più pronunciate in entrambi i grafici, sono l'ExactSHAP e il TreeSHAP, che mostrano prestazioni praticamente identiche. Seguendo questi metodi, c'è il 'KernelSHAP', che presenta una curva meno accentuata ma comunque significativa. Infine, il metodo LIME mostra una curva quasi impercettibile e non comporta alcuna variazione significativa nell'output del modello.

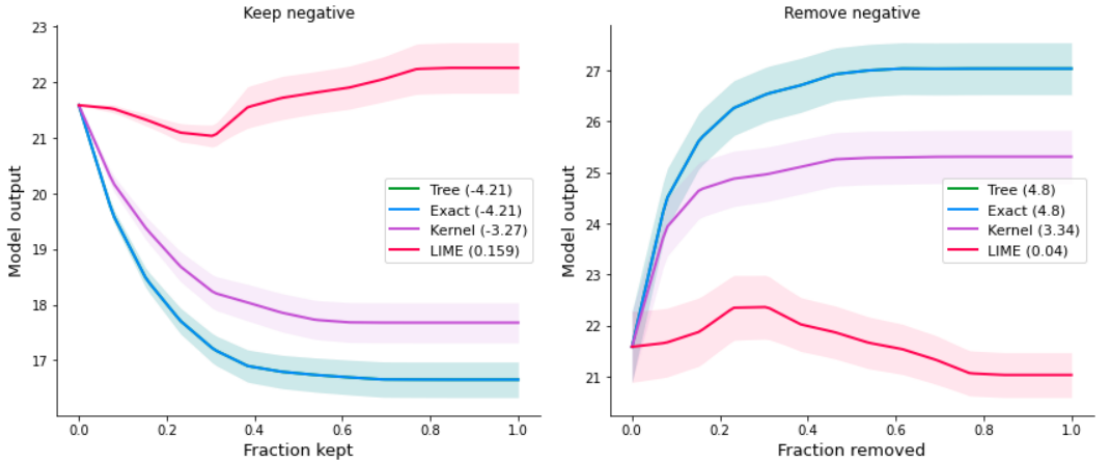


Figura 5.3: Variazione dell'output del modello lungo le metriche "Keep Negative" e "Remove Negative" al variare della frazione tenuta o rimossa

Le stesse osservazioni si applicano al terzo e quarto grafico, dove vengono presentate le stesse caratteristiche. È evidente che le performance migliori sono sempre state ottenute da ExactSHAP e TreeSHAP, seguite da KernelSHAP, e infine da LIME.

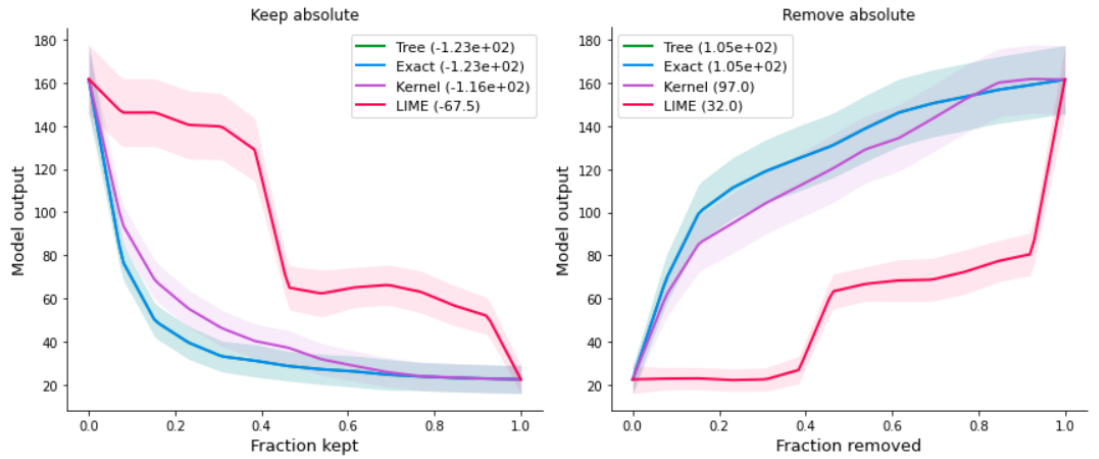


Figura 5.4: Variazione dell'output del modello lungo le metriche "Keep Absolute" e "Remove Absolute" al variare della frazione tenuta o rimossa

Per i grafici "keep absolute" e "remove absolute" viene mostrata la variazione assoluta nell'output del modello. In entrambi i casi, i valori iniziali e finali relativi ad ogni metodo di spiegazione rimangono ovviamente gli stessi. Tuttavia, concentrandoci sulla forma delle curve, possiamo notare che le curve più accentuate e regolari indicano una corretta identificazione delle feature con maggiore influenza in valore assoluto. Questo conferma quanto osservato precedentemente: i metodi di spiegazione più efficaci rimangono ExactSHAP e TreeSHAP, seguiti da KernelSHAP e LIME.

5.2 Esperimento n°2 - Boston RF

Nel secondo esperimento, si è continuato a utilizzare il dataset di **Boston Housing**, già impiegato nel precedente studio. Tuttavia, questa volta è stato adottato l'algoritmo di apprendimento **RandomForest**. La scelta di *RandomForest* è motivata dalla sua capacità intrinseca di assegnare un'importanza alle diverse caratteristiche, che verrà utilizzata come riferimento per valutare le spiegazioni fornite dai vari modelli interpretativi. In questo esperimento, ripeteremo gli stessi test e grafici svolti nel precedente esperimento, ma introdurremo anche la visualizzazione delle importanze assegnate alle feature da ciascun metodo di spiegazione. Inoltre, calcoleremo il **Mean Squared Error** e il **Kendall Tau Error** per confrontare le spiegazioni ottenute dai diversi approcci con quelle fornite dal modello *RandomForest*.

5.2.1 Algoritmi di apprendimento utilizzati

L'algoritmo **Random Forest** è una tecnica di apprendimento automatico che combina l'output di più **alberi decisionali (decision trees)** per migliorare la precisione e la robustezza delle previsioni. È utilizzato per problemi di **classificazione** e **regressione** ed è noto per la sua efficacia nella gestione di overfitting, nella gestione di dataset con molte feature e nella gestione di dati rumorosi.
Ecco come funziona l'algoritmo *Random Forest*:

- **Raccolta dei dati**: si inizia con un dataset che contiene le *variabili di input* (caratteristiche) e le relative *etichette di output* (classi o valori target). Questo dataset viene suddiviso in un *set di addestramento* (training set) e un *set di test* (test set) per valutare le prestazioni dell'algoritmo.
- **Creazione di alberi decisionali**: Per creare un *Random Forest*, vengono generati diversi alberi decisionali. Ogni albero viene addestrato utilizzando un *sottoinsieme casuale* e *bootstrap (campionamento con sostituzione)* del dataset di addestramento. Questo significa che ogni albero è addestrato su un insieme diverso di dati.
- **Divisione dei nodi dell'albero**: Durante la creazione di ciascun albero decisionale, vengono effettuate divisioni nei nodi in base alle caratteristiche che meglio separano i dati. Tuttavia, a differenza di un albero decisionale tradizionale, ogni nodo non considera tutte le caratteristiche, ma solo un sottoinsieme casuale di esse. Questo aiuta a rendere gli alberi più diversificati.
- **Votazione (classificazione) o media (regressione)**: Una volta addestrati tutti gli alberi, quando è necessario effettuare una previsione per un nuovo dato, ciascun albero contribuisce alla decisione finale. Nel caso di problemi di classificazione, ciascun albero "vota" per una classe, e la classe con il maggior numero di voti diventa la previsione del Random Forest. Nel caso di problemi di regressione, ciascun albero restituisce un valore, e la previsione del Random Forest è la media di questi valori.
- **Valutazione delle prestazioni**: Infine, l'algoritmo Random Forest valuta le prestazioni utilizzando il set di test o attraverso tecniche di *cross-validation*. Questo consente di stimare l'accuratezza del modello e di identificare se è presente overfitting.

5.2.2 Risultati numerici

In questo esperimento, è stata adottata una configurazione particolare considerando l'algoritmo di apprendimento, il *Random Forest*. Poiché il *Random Forest* non è nativamente compatibile con l'algoritmo di spiegazione TreeSHAP, è stato invece impiegato un approccio alternativo per l'interpretazione del modello, noto come **TreeApprox**. Il metodo del **"Tree Approximation"** per calcolare lo *Shapley Value* è una tecnica che mira a semplificare il calcolo dei valori di Shapley in situazioni in cui il numero di giocatori o feature è grande e il calcolo diretto dei valori di Shapley potrebbe essere computazionalmente costoso o impraticabile. Questo approccio è spesso utilizzato in contesti di machine learning per attribuire l'importanza delle feature in un modello.
Ecco come funziona in generale il metodo del *"Tree Approximation"* per lo *Shapley Value*:

- **Creazione dell'albero delle feature**: Si inizia creando un albero delle feature, in cui ogni nodo rappresenta una feature o un gruppo di feature. L'obiettivo è suddividere progressivamente le feature in gruppi più piccoli per semplificare il calcolo.

- **Calcolo dei contributi marginali**: Per ogni nodo foglia dell'albero, si calcolano i contributi marginali delle feature contenute in quel gruppo. Il contributo marginale di una feature è l'effetto che ha sulla predizione quando viene aggiunta al gruppo. Questo può essere calcolato utilizzando il modello di machine learning di interesse. Si possono utilizzare diverse metriche, come la differenza nella previsione del modello con e senza la feature.

- **Propagazione dei contributi**: I contributi marginali calcolati nei nodi foglia vengono propagati verso l'alto nell'albero. Questa propagazione può avvenire in diversi modi, a seconda della strategia utilizzata. Una strategia comune è la "bottom-up", in cui i contributi vengono sommati man mano che si risale l'albero.

- **Calcolo dello Shapley Value**: Alla fine del processo, otterrete una stima degli Shapley Values per ciascuna feature. Gli Shapley Values rappresentano l'importanza relativa delle feature nel modello e possono essere utilizzati per interpretare il modello, selezionare le feature più importanti o prenderne decisioni basate sull'importanza delle feature.

Il metodo del *"Tree Approximation"* è una semplificazione del calcolo degli *Shapley value* rispetto all'approccio completo basato sulla permutazione di tutti i giocatori. Tuttavia, l'accuratezza della stima dipende dalla struttura dell'albero delle feature e dalla correttezza del modello di machine learning utilizzato per calcolare i contributi marginali.
In sintesi, il *"Tree Approximation"* per lo *Shapley Value* semplifica il calcolo degli *Shapley Value* attraverso la creazione di un *albero delle feature* e il calcolo dei contributi marginali delle feature in base a questo albero. È particolarmente utile in situazioni in cui il calcolo completo degli Shapley Values sarebbe troppo costoso in termini computazionali.

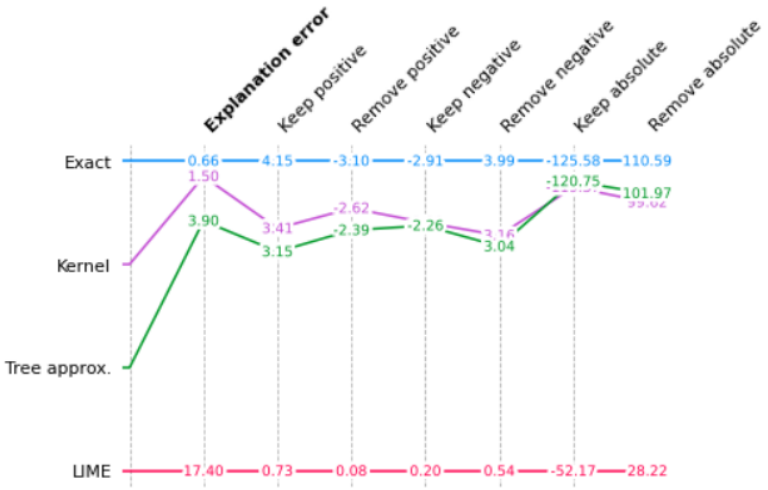


Figura 5.5: Explanation Error e integrale di variazione dell'output di ogni metrica

Come nell'esperimento precedente, in questo grafico 5.5 è evidente che l'algoritmo di spiegazione più efficiente rimane l'*exactSHAP*. Tuttavia, in questa iterazione, in cui è stato sostituito il *treeSHAP* con il *treeApprox*, si nota una performance generalmente inferiore rispetto al *kernelSHAP*. Inoltre, l'algoritmo *LIME* si mantiene costantemente all'ultimo posto in tutte le metriche considerate.

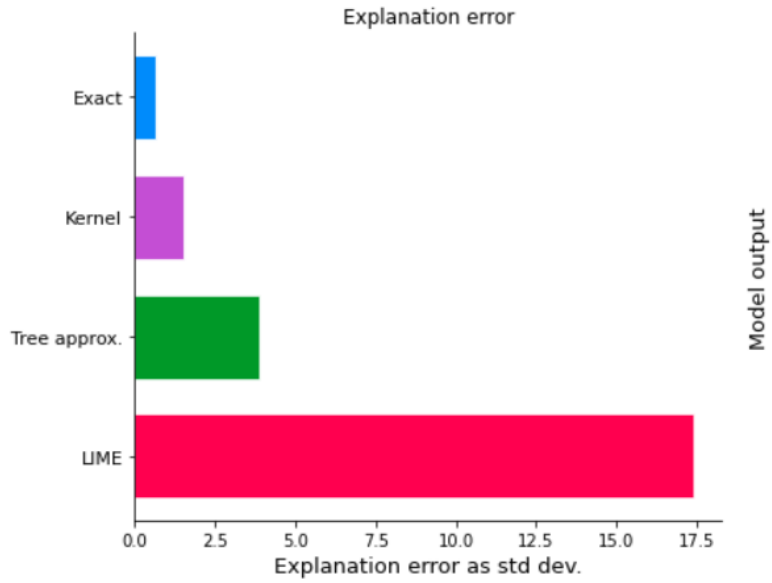


Figura 5.6: Grafico raffigurante tramite istogramma orizzontale la differenza tra gli explanation error dei vari algoritmi

In questo grafico 5.6, si osserva chiaramente che l'errore di spiegazione è minimo nell'approccio ExactSHAP, seguito da KernelSHAP e successivamente da TreeApprox, con LIME che mostra il maggior errore di spiegazione tra tutti gli algoritmi considerati. Questo esperimento dimostra che KernelSHAP è più preciso di TreeApprox, contrariamente a un risultato precedente in cui TreeSHAP aveva superato KernelSHAP in quanto TreeSHAP non è altro che un metodo di calcolo delle importance analogo all'exactSHAP. Questa differenza può essere attribuita alla maggiore efficienza del metodo di campionamento utilizzato in KernelSHAP rispetto a TreeApprox, rendendo KernelSHAP una scelta preferibile quando il numero di feature non permette l'uso di ExactSHAP.

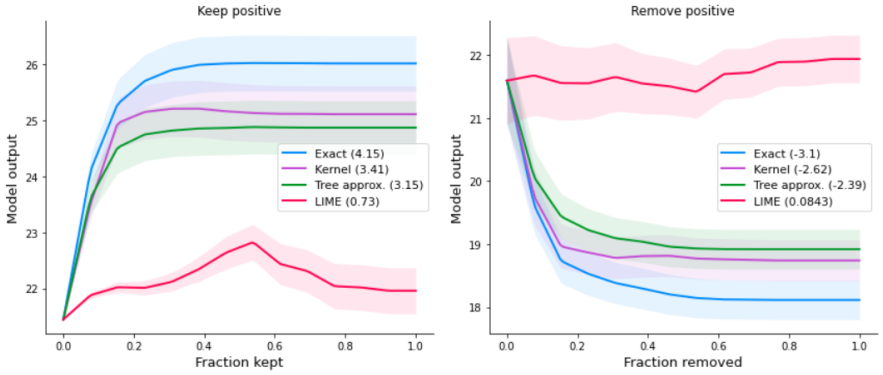


Figura 5.7: Variazione dell'output del modello lungo le metriche "Keep Positive" e "Remove Positive" al variare della frazione tenuta o rimossa

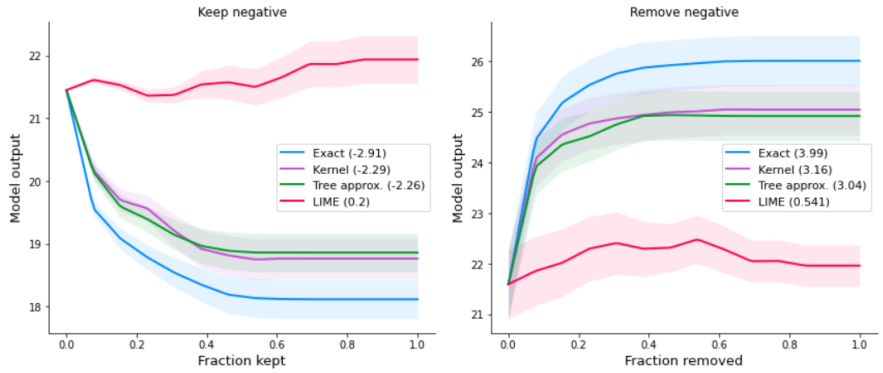


Figura 5.8: Variazione dell'output del modello lungo le metriche "Keep Negative" e "Remove Negative" al variare della frazione tenuta o rimossa

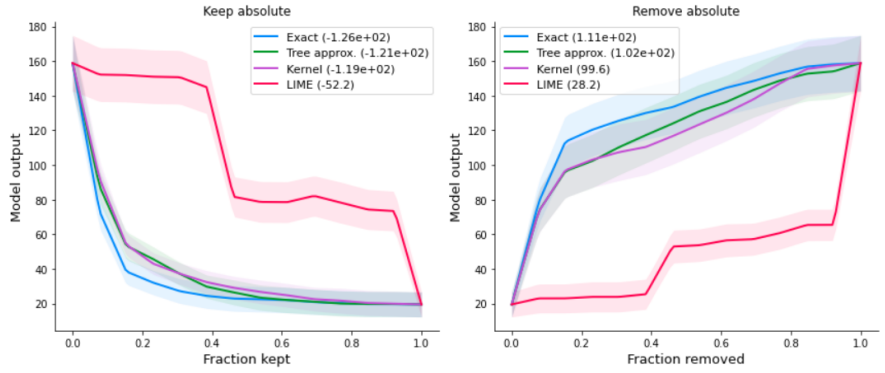


Figura 5.9: Variazione dell'output del modello lungo le metriche "Keep Absolute" e "Remove Absolute" al variare della frazione tenuta o rimossa

Come nel primo esperimento, anche nell'analisi delle variazioni degli output del modello dei tre grafici (5.7,5.8,5.9), l'algoritmo migliore si conferma essere ExactSHAP. Questo emerge chiaramente dalle curve di variazione degli output, che risultano essere più accentuate e regolari. Seguendo la stessa tendenza del grafico precedente, KernelSHAP si posiziona al secondo posto, seguito da TreeApprox, che si colloca quasi alla pari ma con prestazioni leggermente inferiori rispetto a KernelSHAP. Infine, troviamo LIME, caratterizzato da una curva meno regolare e più dispersa rispetto agli altri algoritmi.

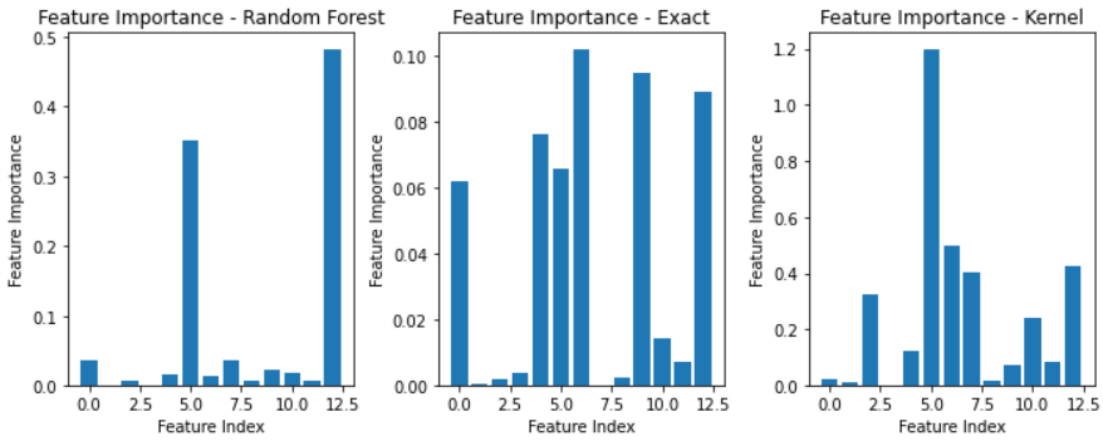


Figura 5.10: Importance attribuite dal learner Random Forest e dai metodi ExactSHAP e KernelSHAP

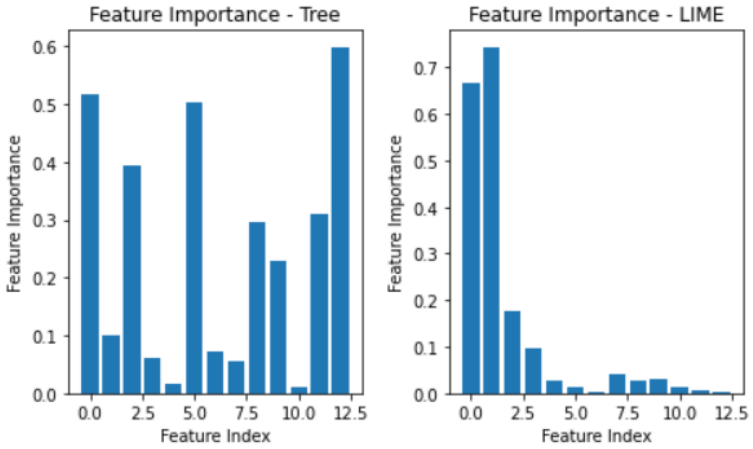


Figura 5.11: Importance attribuite dai metodi TreeApprox e LIME

In queste visualizzazioni (5.10,5.11), sono presentate le valutazioni di importanza assegnate dal modello Random Forest. In questo caso, il modello stesso attribuisce autonomamente l'importanza alle features, fornendo un punto di riferimento per valutare la precisione dei metodi di spiegazione seguenti: ExactSHAP, KernelSHAP, TreeApprox e LIME. Successivamente, verrà calcolato lo scarto di errore tra le attribuzioni generate dal modello Random Forest e quelle ottenute dai metodi di spiegazione, utilizzando metriche come il Mean Squared Error (MSE) e la metrica Kendall Tau per valutare la concordanza tra le valutazioni.
Già da una semplice osservazione visiva, emerge che la feature numero 5 riceve una valutazione di importanza elevata sia dal learner Random Forest che dai metodi SHAP (ExactSHAP, KernelSHAP e TreeApprox). In contrasto, il metodo LIME assegna una valutazione di importanza relativamente bassa a questa stessa feature. Questa discrepanza rappresenta uno dei motivi evidenti per cui si osserva che il metodo LIME non raggiunge un alto grado di precisione nei suoi risultati di spiegazione.

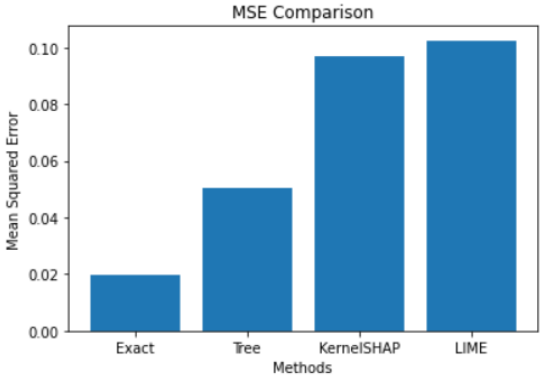


Figura 5.12: Mean Squared Error tra le importance del learner e gli algoritmi di spiegazione

Il grafico sopra 5.12 rappresenta il valore del **Mean Squared Error (MSE)** tra le valutazioni di importanza attribuite dal modello (*Random Forest*) e quelle fornite dai diversi algoritmi di spiegazione. Dato che le caratteristiche intrinseche del dataset sono fisse e la loro influenza sulle etichette di output non dipende dal modello utilizzato, è naturale valutare la precisione degli algoritmi di spiegazione calcolando l'errore quadratico medio tra le loro valutazioni e quelle generate dal modello. In questo contesto, emerge nuovamente la maggiore efficienza complessiva dei metodi SHAP rispetto al metodo LIME. Questa differenza può essere attribuita al fatto che il metodo LIME è model-agnostic, il che significa che ha un collegamento meno stretto con le previsioni specifiche del modello. I valori numerici di tali errori sono:

- **MSE tra Exact e Random Forest**: 0.019
- **MSE tra Tree e Random Forest**: 0.050
- **MSE tra KernelSHAP e Random Forest**: 0.097
- **MSE tra LIME e Random Forest**: 0.102

Mean Squared Error	ExactSHAP	TreeSHAP	KernelSHAP	LIME
RandomForest	0.019	0.050	0.097	0.102

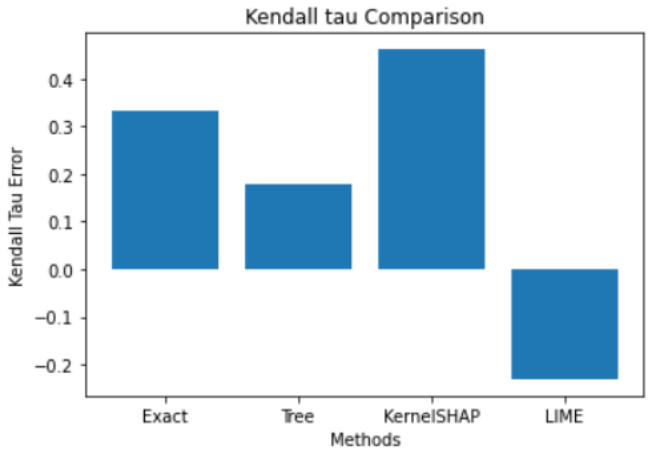


Figura 5.13: Comparazione con la metrica **Kendall Tau** tra le importance del learner e gli algoritmi di spiegazione

La metrica **Kendall Tau** è un indicatore di concordanza o discordanza tra due insiemi di dati ordinati. In questo contesto, viene utilizzata per valutare la concordanza tra le feature importances calcolate da diversi metodi *(ExactSHAP, TreeSHAP, KernelSHAP, LIME*) e quelle ottenute da un modello di *Random Forest (RF).*

Definizione 19 (Metrica Kendall Tau)**.** *La **Kendall Tau**, anche nota come **coefficiente di concordanza di Kendall**, è una misura statistica di concordanza tra due insiemi di dati ordinati. Questa metrica è utilizzata per valutare il grado di associazione o concordanza tra due classificazioni o ordinamenti di dati. Può essere calcolata utilizzando la seguente formula:*

$$\begin{aligned}\tau &= \frac{\textit{numero di coppie concordanti} - \textit{numero di coppie discordanti}}{\textit{numero totale di coppie}} \\ &= 1 - \frac{2(\textit{numero di coppie discordanti})}{\binom{2}{n}}\end{aligned}$$

esprimibile in formula anche come

$$\tau = \frac{2}{n(n-1)} \sum_{i<j} sgn(x_i - x_j) sgn(y_i - y_j)$$

Dove:

- Il **"numero di coppie concordanti"** rappresenta il numero di coppie di elementi che hanno la stessa relazione di ordinamento in entrambi gli insiemi di dati. In altre parole, sono coppie in cui, se un elemento è maggiore di un altro nell'insieme A, lo è anche nell'insieme B, e viceversa.
- Il **"numero di coppie discordanti"** rappresenta il numero di coppie di elementi che hanno relazioni di ordinamento opposte nei due insiemi di dati. In altre parole, sono coppie in cui l'elemento che è maggiore in un insieme è minore nell'altro e viceversa.
- Il **"numero totale di coppie"** è semplicemente il numero totale di possibili coppie di elementi nei due insiemi di dati.
- $\binom{2}{n} = \frac{n(n-1)}{2}$ è il coefficiente binomiale della scelta di 2 oggetti in un insieme di n oggetti

Una volta calcolato il valore di τ, questo può variare tra -1 e 1:

- Se $\tau = 1$, significa che c'è una perfetta concordanza tra i due insiemi di dati. Ogni coppia di elementi è **concorde**.
- Se $\tau = -1$, significa che c'è una perfetta discordanza tra i due insiemi di dati. Ogni coppia di elementi è **discordante**.
- Se $\tau = 0$, significa che non c'è **né concordanza né discordanza** tra i due insiemi di dati. Le coppie concordanti e discordanti sono bilanciate.

Il valore di Tau è particolarmente utile quando si desidera valutare la concordanza tra le classificazioni o gli ordinamenti senza preoccuparsi dei valori assoluti dei dati, ma solo delle loro relazioni di ordinamento.

I risultati riportati mostrano i valori della metrica **Kendall Tau** per ciascun metodo rispetto alla *Random Forest*:

- **Exact vs RF**: $\tau = 0.333$
  Questo valore positivo indica una moderata concordanza tra le feature importances calcolate tramite il metodo ExactSHAP e quelle della Random Forest. Ciò suggerisce che le feature considerate importanti secondo il metodo ExactSHAP tendono ad essere considerate anche importanti dalla Random Forest.

- **Tree vs RF**: $\tau = 0.179$
  Questo valore positivo indica una leggera concordanza tra le feature importances calcolate tramite il metodo TreeSHAP e quelle della Random Forest. Anche se la concordanza è meno pronunciata rispetto al metodo ExactSHAP, suggerisce comunque una certa coerenza tra le due metodologie.

- **Kernel vs RF**: $\tau = 0.461$
  Questo valore positivo indica una concordanza relativamente forte tra le feature importances calcolate tramite il metodo KernelSHAP e quelle della Random Forest. La concordanza più elevata rispetto agli altri metodi suggerisce che le feature considerate importanti da KernelSHAP tendono ad essere considerate anche importanti dalla Random Forest.

- **LIME vs RF**: $\tau = -0.153$
  Questo valore negativo indica una discordanza tra le feature importances calcolate tramite il metodo LIME e quelle della Random Forest. Ciò suggerisce che le feature considerate importanti da LIME possono essere considerate meno importanti o addirittura invertite in importanza dalla Random Forest. Questa discordanza potrebbe essere dovuta alle differenze di approccio e di modellazione tra LIME e la Random Forest.

Kendall Tau	ExactSHAP	TreeSHAP	KernelSHAP	LIME
RandomForest	0.333	0.179	0.461	-0.153

5.3 Esperimento n°3 - Boston RF perturbato

Questo esperimento è stato condotto in modo simile a quello precedente, dove infatti si utilizza come dataset il *Boston Housing* e come learner la *Random Forest*, con la differenza che questa volta il dataset è stato alterato introducendo delle variazioni casuali ai dati. In particolare, ogni punto nel dataset è stato modificato aggiungendo un valore casuale proveniente da una distribuzione normale con media (loc) 0 e una deviazione standard (scale) di 0.1. Questo processo ha introdotto una sorta di "rumore" nei dati, causando delle piccole variazioni casuali in ciascun punto del dataset. Dopo questa perturbazione, i valori dei dati sono stati limitati (clip) nell'intervallo compreso tra 0 e 1 per assicurarsi che rimanessero all'interno di un range valido. L'obiettivo di questa procedura è testare come gli algoritmi di spiegazione si comportano quando i dati non sono puliti e presentano perturbazioni di vario genere.

5.3.1 Risultati numerici

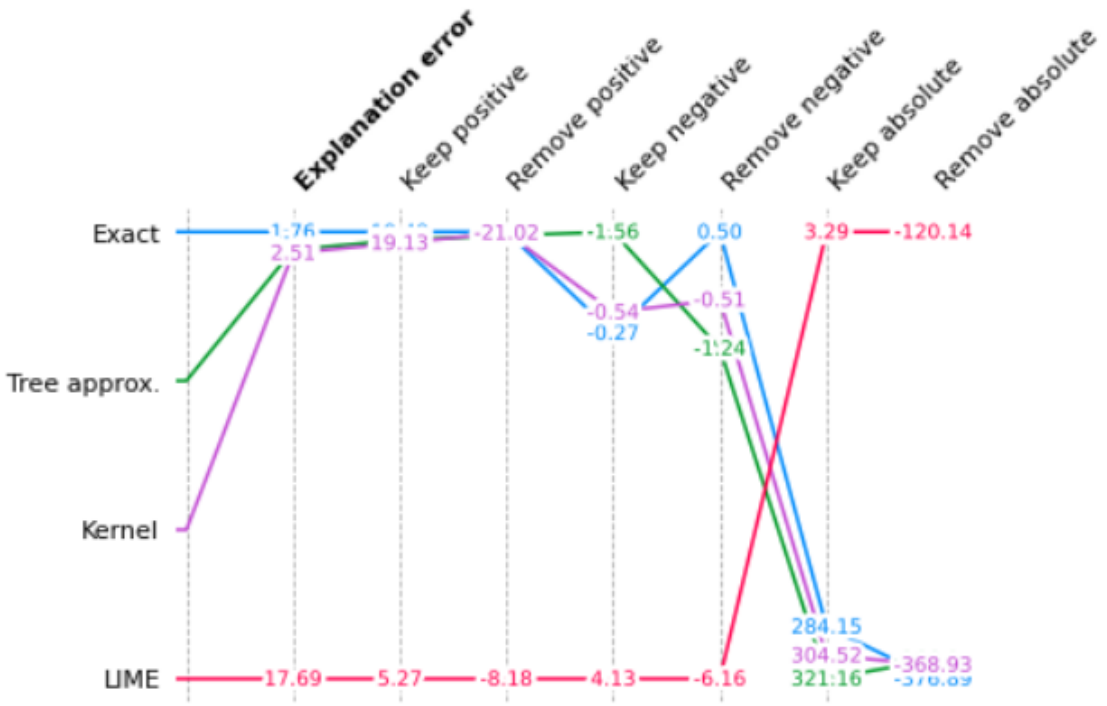


Figura 5.14: Explanation Error e integrale di variazione dell'output di ogni metrica

Da questo grafico, è evidente un generale deterioramento delle prestazioni a causa della perturbazione dei dati (infatti, Exact passa da un explanation error di 0.66 a 1.76). Tuttavia, è interessante notare che l'ordine generale degli algoritmi tende a rimanere più o meno lo stesso, con ExactSHAP, TreeApprox, KernelSHAP e LIME mantenendo posizioni simili.
Tuttavia, vi sono alcune eccezioni notevoli. Ad esempio, nella metrica "Keep negative," TreeApprox emerge come il miglior algoritmo, seguito da KernelSHAP e ExactSHAP. Allo stesso modo, nelle metriche "Keep Absolute" e "Remove Absolute," LIME sorprendentemente ottiene i risultati migliori. Questo fenomeno potrebbe essere attribuito alla diversa sensibilità delle metriche alle perturbazioni, specialmente nel caso delle metriche "Absolute," in cui le perturbazioni possono avere un segno variabile.

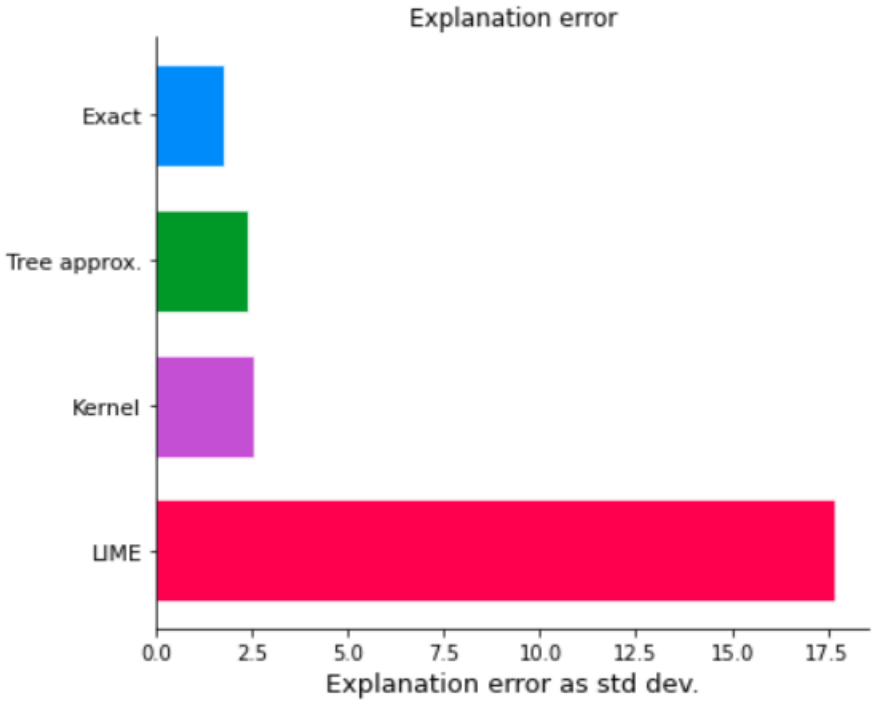


Figura 5.15: Grafico raffigurante tramite istogramma orizzontale la differenza tra gli explanation error dei vari algoritmi

In questo secondo esperimento, osserviamo che il posizionamento relativo degli errori di spiegazione rimane invariato, tuttavia, si evidenzia un notevole aumento nell'errore di spiegazione associato a LIME.

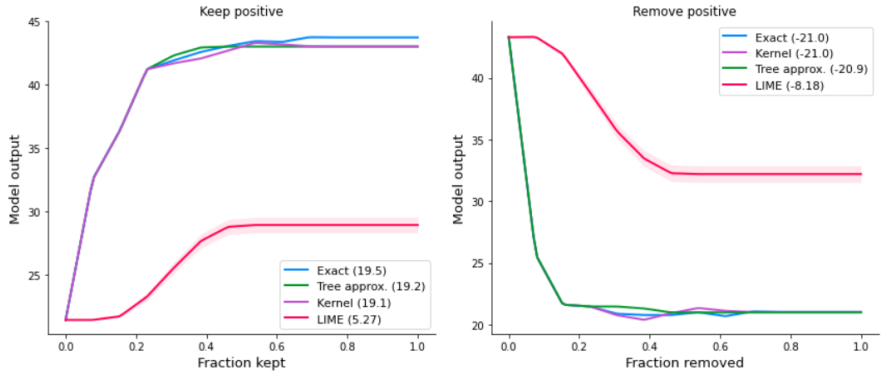


Figura 5.16: Variazione dell'output del modello lungo le metriche "Keep Positive" e "Remove Positive" al variare della frazione tenuta o rimossa

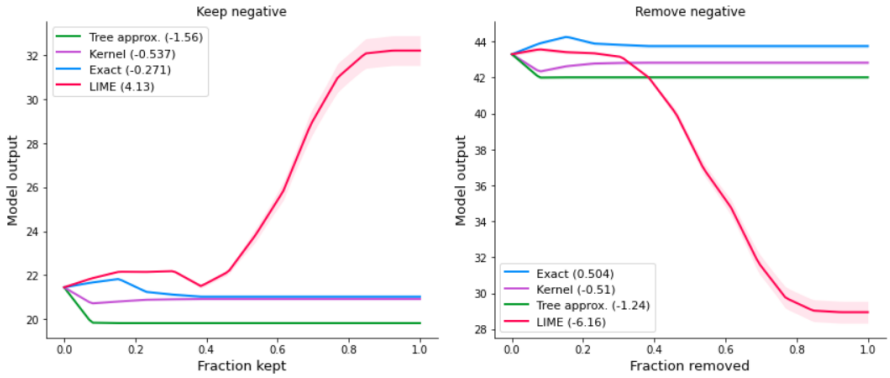


Figura 5.17: Variazione dell'output del modello lungo le metriche "Keep Negative" e "Remove Negative" al variare della frazione tenuta o rimossa

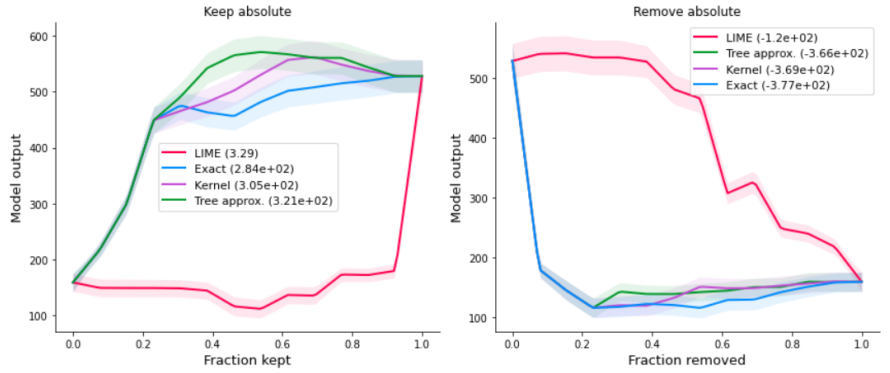


Figura 5.18: Variazione dell'output del modello lungo le metriche "Keep Absolute" e "Remove Absolute" al variare della frazione tenuta o rimossa

Come evidente dall'analisi visuale dei tre grafici, l'introduzione delle perturbazioni ha notevolmente influenzato la stabilità delle curve nella variazione del model output nelle metriche di Keep/Remove Negative, Keep/Remove Positive e Keep/Remove Absolute. Le perturbazioni hanno reso le curve molto più irregolari e oscillanti.
Inoltre, è immediatamente evidente che le prestazioni generali hanno subito un significativo deterioramento, soprattutto nei casi di Keep/Remove Negative e Keep/Remove Absolute. Questo declino nelle prestazioni è particolarmente evidente nei metodi SHAP, che sembrano essere notevolmente influenzati dalle perturbazioni. Questi risultati suggeriscono che i metodi SHAP possono essere altamente efficaci quando si lavora con dati puliti e stabili nel dataset, mentre possono risentire negativamente di situazioni in cui i dati sono soggetti a perturbazioni.

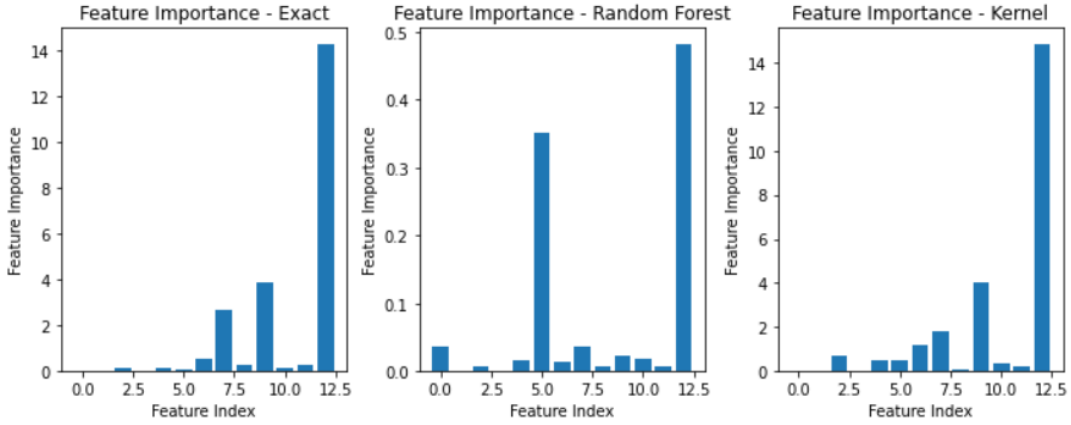


Figura 5.19: Importance attribuite dal learner Random Forest e dai metodi ExactSHAP e KernelSHAP

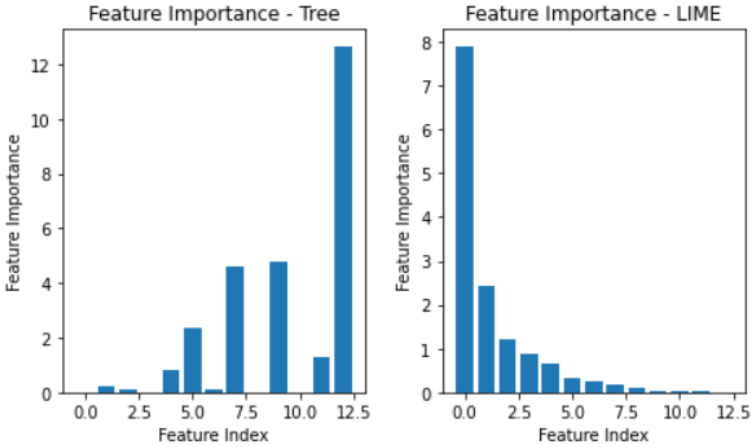


Figura 5.20: Importance attribuite dai metodi TreeApprox e LIME

Come già osservato nei precedenti esperimenti, anche in questo caso ci sono feature le cui importanze assegnate dal learner e quelle calcolate dai metodi SHAP risultano simili. Ad esempio, possiamo notare che la feature 12 è classificata come "significativa" in tutti gli algoritmi SHAP. Tuttavia, a differenza degli esperimenti precedenti, si osserva una maggiore varianza tra le feature in questo contesto.
D'altra parte, nel caso di LIME, si osservano prestazioni relativamente stabili e simili, suggerendo una possibile maggiore robustezza alle perturbazioni rispetto ai metodi SHAP.

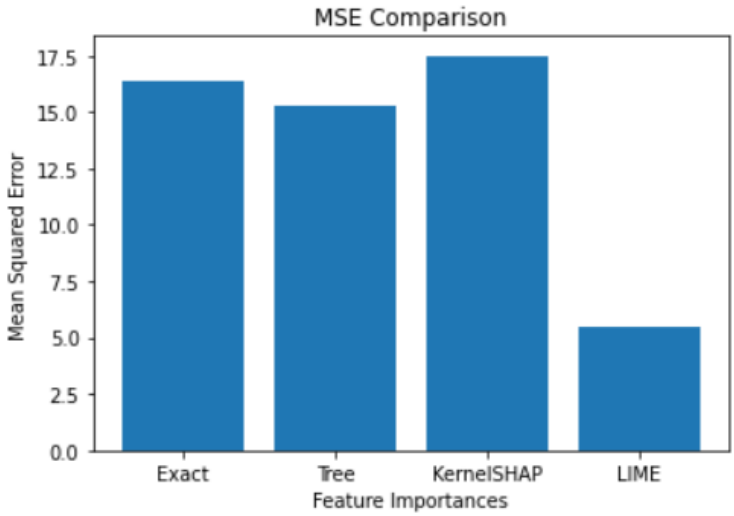


Figura 5.21: Mean Squared Error tra le importance del learner e gli algoritmi di spiegazione

Mean Squared Error	ExactSHAP	TreeSHAP	KernelSHAP	LIME
RandomForest	16.362	15.270	17.496	5.440

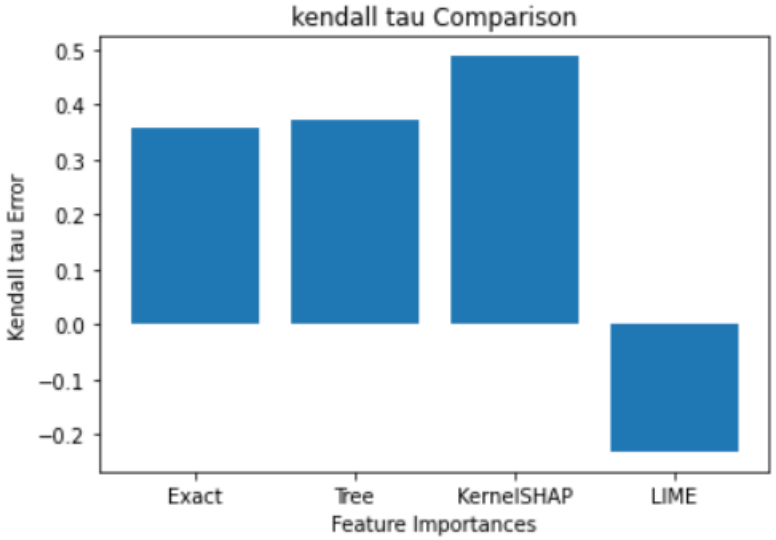


Figura 5.22: Comparazione con la metrica **Kendall Tau** tra le importance del learner e gli algoritmi di spiegazione

Kendall Tau	ExactSHAP	TreeSHAP	KernelSHAP	LIME
RandomForest	0.358	0.373	0.487	-0.230

Come era prevedibile, in questo contesto gli MSE tra le spiegazioni fornite dai metodi SHAP e quelle del Random Forest si sono notevolmente ampliati rispetto all'esperimento precedente, superando persino l'errore dell'algoritmo LIME. Tuttavia, è interessante notare che, nonostante l'aumento dell'errore, la metrica Kendall Tau mostra ancora una correlazione positiva tra le spiegazioni dei metodi SHAP. Questo suggerisce che, in generale, i metodi SHAP continuano a attribuire importanza alle stesse feature, nonostante l'ampiezza dell'errore nelle spiegazioni.

5.4 Esperimento n°4 - Adult dataset XGBoost

In questo nuovo esperimento, è stato utilizzato utilizzato il dataset "adult", precedentemente estratto da Barry Becker dal database del censimento del 1994. È importante notare che sono state applicate condizioni rigorose per estrarre un insieme di dati ragionevolmente puliti, basate su criteri come l'età, il guadagno e le ore lavorate, con l'obiettivo di predire se una persona guadagna più di 50.000 dollari l'anno.
A differenza dell'esperimento precedente, in questa occasione ci si è concentrati su un problema di classificazione, anziché di regressione, al fine di valutare le prestazioni di vari metodi di spiegazione. Inoltre, è stato utilizzato l'algoritmo di apprendimento XGBoost come learner principale.

5.4.1 Dataset

Il dataset "Adult" rappresenta un'ampia raccolta di dati relativi a individui e presenta una varietà di attributi che riflettono diverse caratteristiche socio-demografiche. L'obiettivo principale di questo dataset è prevedere se un individuo guadagna più di 50.000 dollari all'anno (">50K") o meno ("<=50K") in base alle informazioni fornite. Ecco una breve panoramica degli attributi chiave presenti nel dataset:

- **Età (age):** Questo attributo rappresenta l'età continua di un individuo.
- **Classe Lavorativa (workclass):** Indica la classe lavorativa dell'individuo e può essere Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay o Never-worked.
- **Peso FNLWGT (fnlwgt):** Questo è un valore continuo che rappresenta il peso ponderato dei dati e può essere utilizzato per rappresentare la popolazione.
- **Istruzione (education):** Indica il livello di istruzione dell'individuo, con categorie come Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th o Preschool.
- **Anni di Istruzione (education-num):** Rappresenta il numero di anni di istruzione completati in modo continuo.
- **Stato Civile (marital-status):** Indica lo stato civile dell'individuo, come Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent o Married-AF-spouse.

- **Occupazione (occupation):** Specifica la professione o l'occupazione dell'individuo, con categorie come Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv o Armed-Forces.
- **Razza (race):** Rappresenta l'appartenenza etnica dell'individuo e può essere White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other o Black.
- **Sesso (sex):** Specifica il genere dell'individuo come Female o Male.
- **Guadagno in Capitale (capital-gain):** Questo attributo rappresenta i guadagni in capitale dell'individuo in forma continua.
- **Perdita in Capitale (capital-loss):** Rappresenta le perdite in capitale dell'individuo in forma continua.
- **Ore Lavorative Settimanali (hours-per-week):** Indica il numero di ore lavorate da un individuo ogni settimana, rappresentato in forma continua.
- **Paese d'Origine (native-country):** Specifica il paese d'origine dell'individuo, con una vasta gamma di paesi, tra cui gli Stati Uniti (United-States), ma anche altri paesi come India, Giappone, Messico, Italia e molti altri.

Questo dataset è ampiamente utilizzato per scopi di classificazione e analisi predittiva, consentendo di esplorare le relazioni tra le caratteristiche socio-demografiche degli individui e il loro reddito annuale.

5.4.2 Risultati

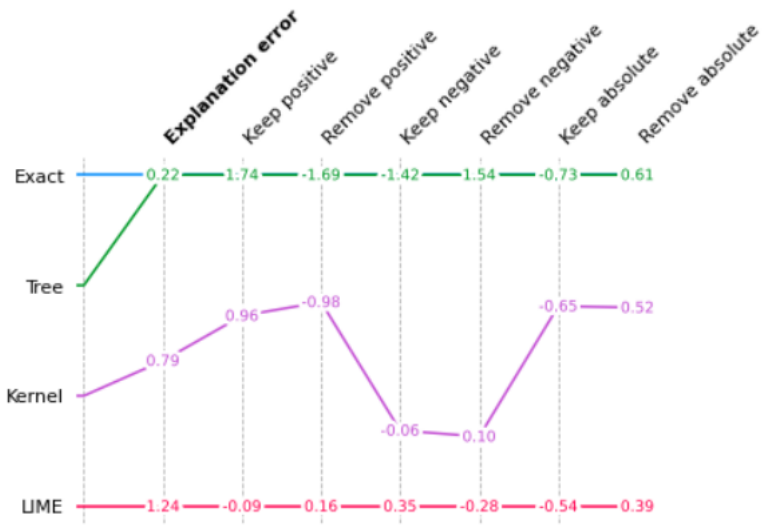


Figura 5.23: Explanation Error e integrale di variazione dell'output di ogni metrica

Come nel caso della regressione, emerge chiaramente che i due metodi di spiegazione più accurati sono l'ExactSHAP e il TreeSHAP, seguiti dal KernelSHAP, mentre il LIME risulta essere meno preciso in confronto. Questo risultato evidenzia che l'efficienza dei metodi menzionati si mantiene anche in un problema di classificazione, confermando la loro affidabilità in diverse situazioni.
Inoltre, è interessante notare che la precisione dei metodi di spiegazione si mantiene sia quando si lavora con feature numeriche che con feature categoriche, dimostrando la loro utilità e versatilità in vari contesti di analisi dei dati.

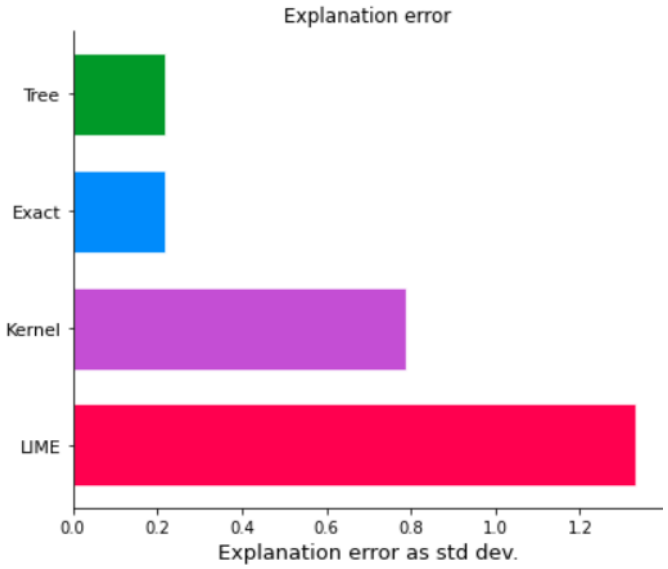


Figura 5.24: Grafico raffigurante tramite istogramma orizzontale la differenza tra gli explanation error dei vari algoritmi

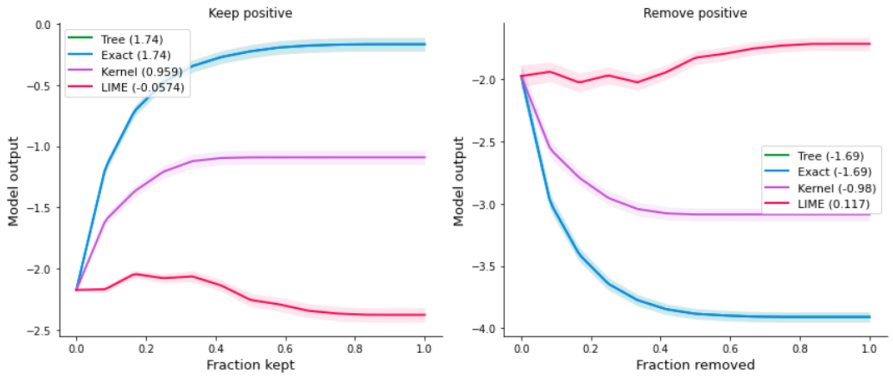


Figura 5.25: Variazione dell'output del modello lungo le metriche "Keep Positive" e "Remove Positive" al variare della frazione tenuta o rimossa

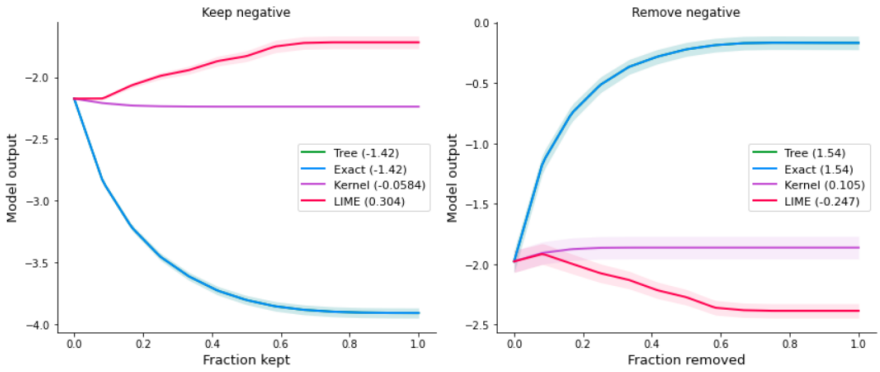


Figura 5.26: Variazione dell'output del modello lungo le metriche "Keep Negative" e "Remove Negative" al variare della frazione tenuta o rimossa

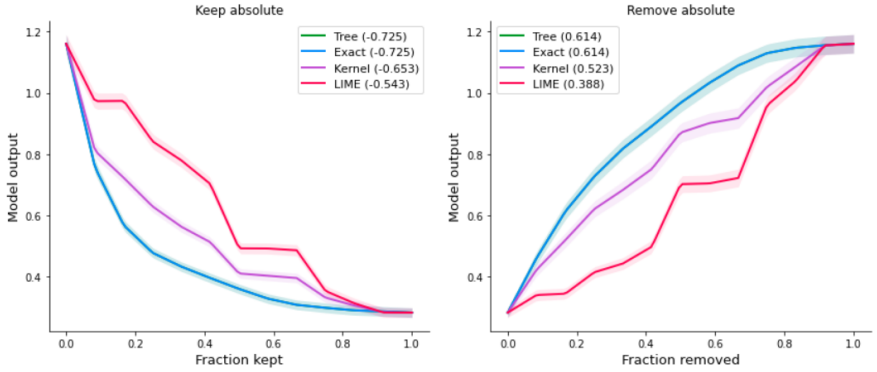


Figura 5.27: Variazione dell'output del modello lungo le metriche "Keep Absolute" e "Remove Absolute" al variare della frazione tenuta o rimossa

Come precedentemente anticipato, le prestazioni dei metodi di spiegazione in questo problema di classificazione seguono un trend simile a quanto osservato negli esperimenti di regressione. Infatti, nei grafici relativi alle varie metriche di "Keep/Remove" i metodi che mostrano le migliori performance, evidenziate da curve più lisce, sono ancora una volta *ExactSHAP* e *TreeSHAP*, seguiti da *KernelSHAP* e *LIME*.

5.5 Esperimento n°5 - Adult XGBoost perturbato

In questo esperimento più recente, sono stati applicati i metodi di spiegazione al dataset "Adult" utilizzato in precedenza ma, in questa occasione, sono state introdotte delle perturbazioni alle feature non categoriche. L'obiettivo è valutare le prestazioni degli algoritmi di spiegazione in un contesto di classificazione che coinvolge dati non puliti o soggetti a perturbazioni. In questa situazione, le metriche *"keep/remove"* sono state utilizzate per quantificare la variazione nel numero di output classificati come "$> 50k$" e "$< 50k$". In

particolare, è stato assegnato il valore +1 al primo caso e −1 al secondo caso nelle metriche di "keep/remove".

5.5.1 Risultati

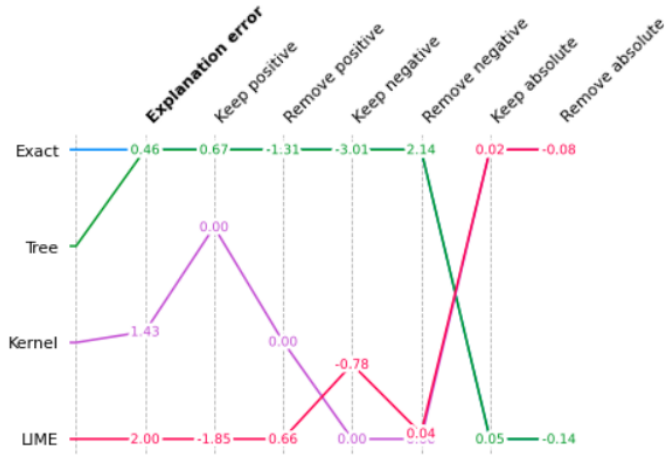


Figura 5.28: Explanation Error e integrale di variazione dell'output di ogni metrica

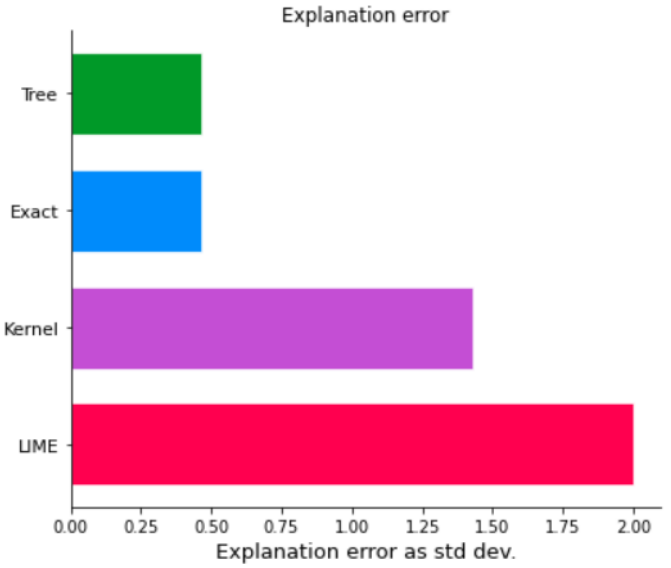


Figura 5.29: Grafico raffigurante tramite istogramma orizzontale la differenza tra gli explanation error dei vari algoritmi

Ancora una volta, in questo recente esperimento, abbiamo osservato che i migliori algoritmi di spiegazione sono stati l'ExactSHAP e il TreeSHAP. Nonostante abbiano subito un

notevole deterioramento nella precisione a causa delle perturbazioni, hanno mantenuto la loro posizione di vertice in termini di Explanation Error e nelle metriche "Keep/Remove Positive/Negative".
Tuttavia, la situazione è differente per le metriche "Keep/Remove Absolute" che, come osservato nei precedenti esperimenti perturbati, hanno registrato un marcato decadimento delle performance. In queste metriche, gli algoritmi SHAP sono risultati essere persino meno efficaci dell'algoritmo LIME.
Per quanto riguarda il metodo KernelSHAP, è emerso un notevole peggioramento delle prestazioni dovuto sia alle perturbazioni che al suo metodo di campionamento.
In particolare, KernelSHAP si è rivelato meno efficace di LIME nella maggior parte delle metriche "Keep/Remove" mantenendo un vantaggio solo nell'Explanation Error. Questo deterioramento è stato causato dalle perturbazioni, che hanno influenzato le importance attribuite alle feature, rendendo difficile distinguere quelle che contribuiscono positivamente all'output del modello da quelle che contribuiscono negativamente. Questa difficoltà si riflette nei valori di 0 nelle metriche "Keep/Remove Positive/Negative" per KernelSHAP.

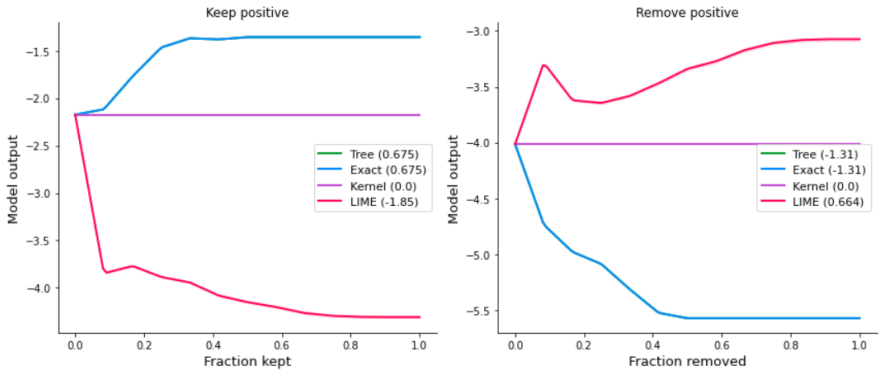


Figura 5.30: Variazione dell'output del modello lungo le metriche "Keep Positive" e "Remove Positive" al variare della frazione tenuta o rimossa

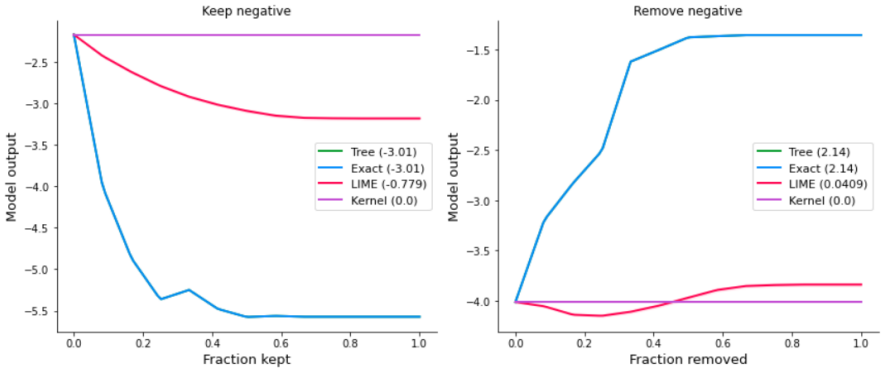


Figura 5.31: Variazione dell'output del modello lungo le metriche "Keep Negative" e "Remove Negative" al variare della frazione tenuta o rimossa

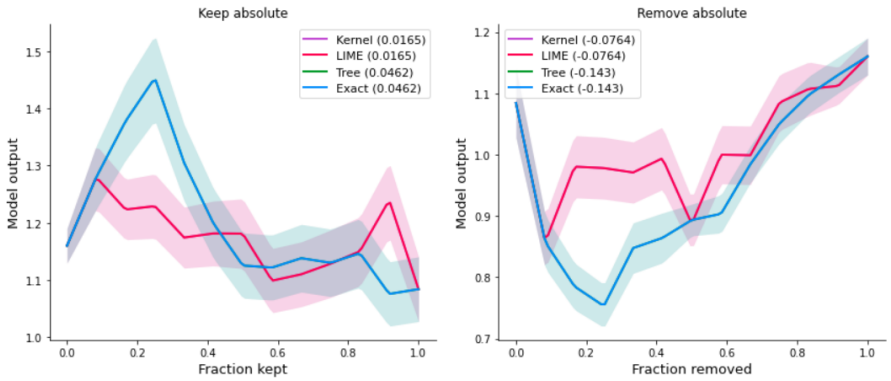


Figura 5.32: Variazione dell'output del modello lungo le metriche "Keep Absolute" e "Remove Absolute" al variare della frazione tenuta o rimossa

Come precedentemente evidenziato, i deterioramenti nelle performance sono chiaramente riscontrabili anche attraverso l'analisi dei grafici che mostrano l'influenza delle varie metriche "Keep/Remove." Nel caso del KernelSHAP, questi decadimenti sono visibili in tutte le metriche, mentre per ExactSHAP e TreeSHAP, si notano principalmente nelle metriche "Absolute." Tuttavia, è importante sottolineare che anche in altre metriche, le curve risultano meno accentuate e meno precise rispetto all'esperimento precedente.
Nel caso di LIME, la differenza non è così ampia poiché LIME si concentra sull'analisi delle singole feature separatamente. Questo approccio ha reso LIME meno sensibile all'influenza delle coalizioni delle feature, risultando paradossalmente più stabile, seppur meno preciso.

5.6 Esperimento n°6 - correlazione tra feature

In questo esperimento viene esplorato il funzionamento dei metodi di spiegazione in presenza di dati con diverse caratteristiche di correlazione. A tale scopo, sono stati generati due dataset distinti: uno con dati aventi feature scarsamente correlate e l'altro con feature altamente correlate. Entrambi i dataset sono composti da 1000 campioni, ciascuno caratterizzato da 10 feature. Nel primo dataset, le feature sono state generate da una distribuzione normale sferica con media zero e matrice di covarianza uguale all'identità. Nel secondo dataset, invece, sono state introdotte correlazioni casuali tra le feature.
L'obiettivo principale di questo esperimento è quello di applicare un learner ai due dataset e successivamente implementare i metodi di spiegazione al fine di analizzare se le prestazioni di tali metodi siano influenzate dalla presenza o dall'assenza di correlazioni tra le feature. Per entrambi i dataset, si considera un vettore di pesi predefinito $([2, -1, 0.5, 1.5, 0, -0.5, 1, -2, 0, 0.5])$ per la generazione dell'output, in modo da mantenere una costante per la relazione tra le feature di input e l'output, consentendo così una valutazione accurata dei metodi di spiegazione.
Questo esperimento riveste una notevole importanza poiché permette di analizzare se la presenza di correlazioni tra le feature influisca sulla capacità dei metodi di spiegazione

di fornire interpretazioni accurate e affidabili dei modelli. I risultati ottenuti da questa analisi contribuiscono significativamente alla comprensione delle dinamiche di spiegazione nei contesti di dati con differenti gradi di correlazione, avanzando così nell'interpretabilità dei modelli di machine learning.

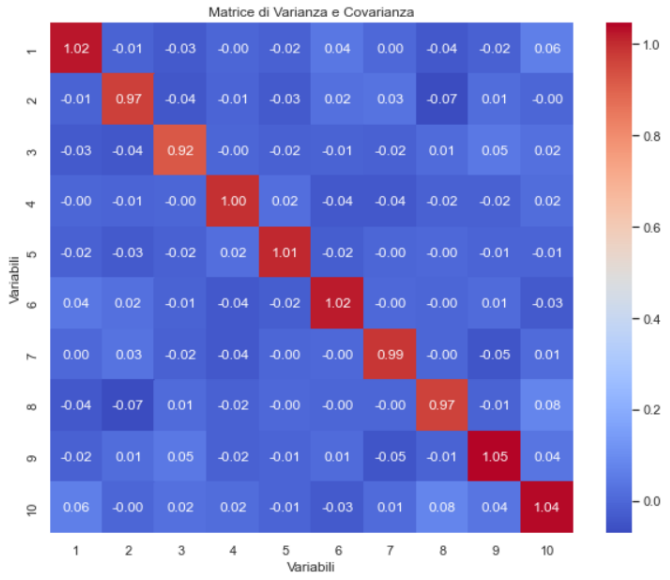


Figura 5.33: Matrice di Varianza e Covarianza del primo dataset

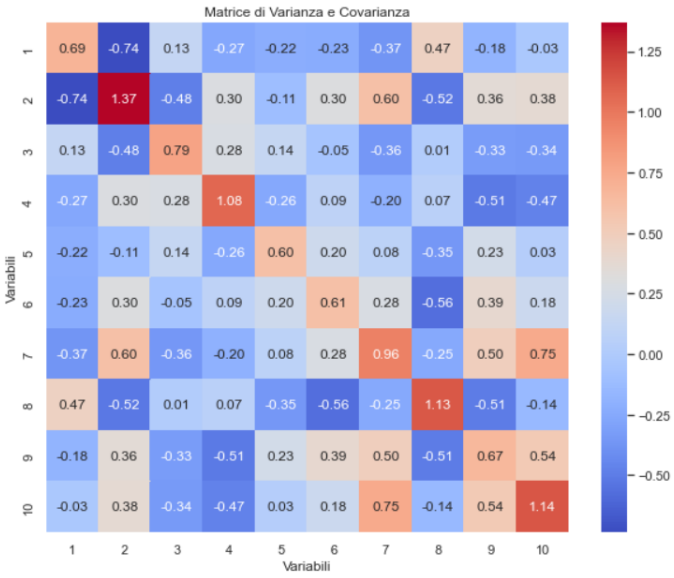
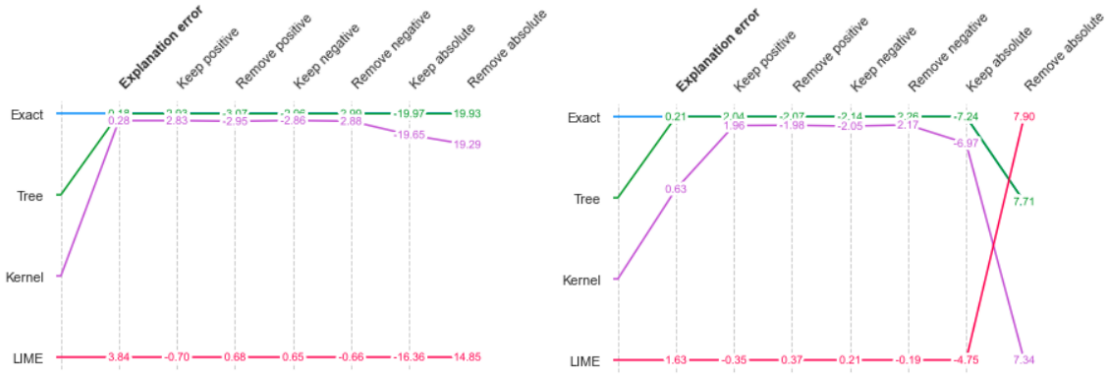


Figura 5.34: Matrice di Varianza e Covarianza del secondo dataset

Il primo grafico sopra riportato 5.33 rappresenta la matrice di varianza e covarianza dei dati campionati dal primo dataset. Questa visualizzazione evidenzia chiaramente che le feature presenti nel primo dataset mostrano correlazioni di modulo molto basso, con valori prossimi allo zero. Ciò indica che le feature del primo dataset sono praticamente indipendenti l'una dall'altra, suggerendo una bassa interdipendenza tra di esse. Il secondo grafico 5.34 invece illustra la matrice di varianza e covarianza dei dati campionati dal secondo dataset. È evidente dalla rappresentazione che le feature in questo secondo dataset presentano correlazioni con modulo significativamente maggiore di zero. Questo indica che le feature nel secondo dataset sono correlate tra di loro, mostrando una maggiore interdipendenza rispetto al primo dataset. La presenza di correlazioni più pronunciate tra le feature può avere un impatto significativo sulle analisi e sulle interpretazioni dei dati e dei modelli applicati a essi.

5.6.1 Risultati

Nel contesto di questo esperimento, è stato adottato un approccio basato su un regressore *XGBoost* come modello di apprendimento automatico (learner). L'obiettivo principale è stato quello di valutare le performance dei metodi di spiegazione utilizzati precedentemente in presenza di dataset caratterizzati da differenti gradi di correlazione tra le feature. Questa analisi mira a rispondere alla fondamentale domanda se le prestazioni di tali metodi di spiegazione siano influenzate dalla presenza o dall'assenza di correlazioni tra le feature dei dati campionati.
Per condurre questa valutazione, sono stati applicati gli stessi metodi di valutazione e metriche utilizzati nei precedenti esperimenti a entrambi i dataset. Ciò consentirà di confrontare direttamente le performance dei metodi di spiegazione in contesti con correlazioni trascurabili e non trascurabili tra le feature dei dati. L'analisi dei risultati ottenuti sarà cruciale per comprendere se e come la struttura delle correlazioni tra le feature influisca sulla capacità dei metodi di spiegazione di fornire interpretazioni accurate e affidabili dei modelli di machine learning.

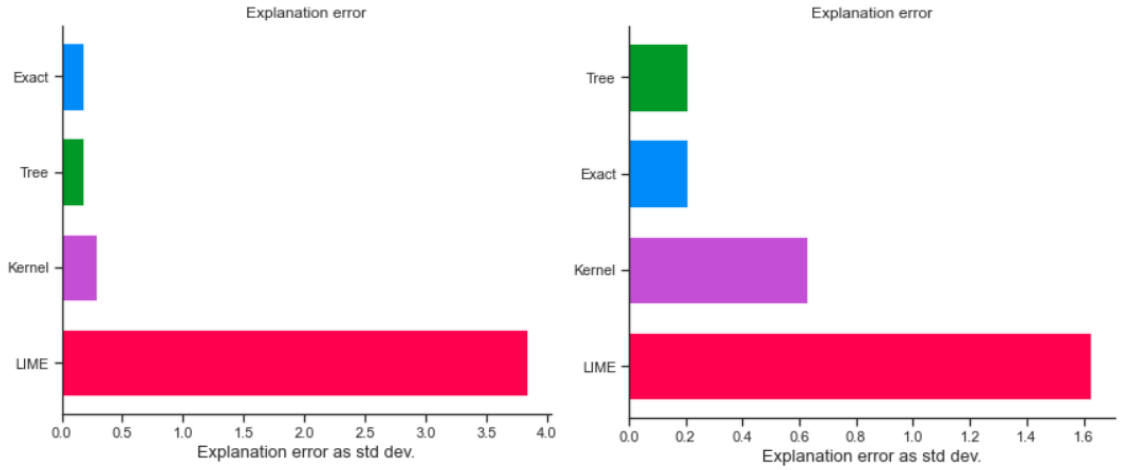


(a) Valori delle metriche nel caso del primo dataset

(b) Valori delle metriche nel caso del secondo dataset

Figura 5.35: Comparazione performance dei metodi nei due dataset

Una prima osservazione evidente è che, nel caso degli algoritmi SHAP, si manifesta un deciso deterioramento delle performance in tutte le metriche quando le feature presentano una correlazione significativa. In contrasto, nel caso di LIME, si osserva un effetto opposto: l'algoritmo migliora le performance in tutte le metriche. Questo fenomeno potrebbe essere spiegato dalla differenza nell'approccio adottato da questi due tipi di algoritmi.
Gli algoritmi SHAP si basano sull'assunzione che le feature siano indipendenti tra di loro, e quindi tendono a ottenere risultati più precisi quando la correlazione tra le feature è minima. D'altra parte, LIME sembra trarre vantaggio dalla presenza di correlazioni, poiché si concentra sull'importanza delle singole feature senza considerare il loro contributo all'interno di combinazioni più complesse.
Questi risultati evidenziano l'importanza di considerare attentamente la struttura delle correlazioni tra le feature quando si scelgono gli algoritmi di spiegazione da utilizzare.



(a) Explanation error nel primo dataset (b) Explanation error nel secondo dataset

Figura 5.36: Comparazione dei metodi nei due dataset

I due grafici presentati in dettaglio mettono in evidenza la differenza nelle performance, valutate in base all'errore di spiegazione, tra il dataset con feature poco correlate e quello con feature altamente correlate. L'analisi di questi grafici rivela una tendenza chiara e significativa: nel caso di LIME, l'errore di spiegazione diminuisce notevolmente quando le feature presentano una correlazione maggiore, mentre per gli algoritmi SHAP si osserva un aumento dell'errore di spiegazione in situazioni simili.
Questi risultati sottolineano il comportamento opposto di LIME e degli algoritmi SHAP rispetto alla presenza di correlazioni tra le feature. In particolare, LIME sembra beneficiare della correlazione tra le feature, riducendo l'errore di spiegazione, mentre gli algoritmi SHAP mostrano una maggiore difficoltà nell'interpretare dati con feature fortemente correlate, il che si traduce in un aumento dell'errore di spiegazione.
.

5.7 Esperimento n°7 - correlazione tra feature

Nell'ambito dell' indagine sulla comprensione dell'efficienza dei metodi di spiegazione in contesti di dati con differenti gradi di correlazione tra le feature, è proposto un nuovo esperimento. Tuttavia, a differenza dell'esperimento precedente, in questo caso è stato adottato un algoritmo di apprendimento differente, ovvero la Random Forest.
Come nell'esperimento precedente, l'obiettivo primario rimane lo stesso: valutare l'efficienza dei vari metodi di spiegazione variando il livello di correlazione tra le feature dei dati campionati. L'analisi si concentrerà sulle performance di questi metodi, cercando di comprendere come la loro efficacia sia influenzata dalla presenza o dall'assenza di correlazioni tra le feature.

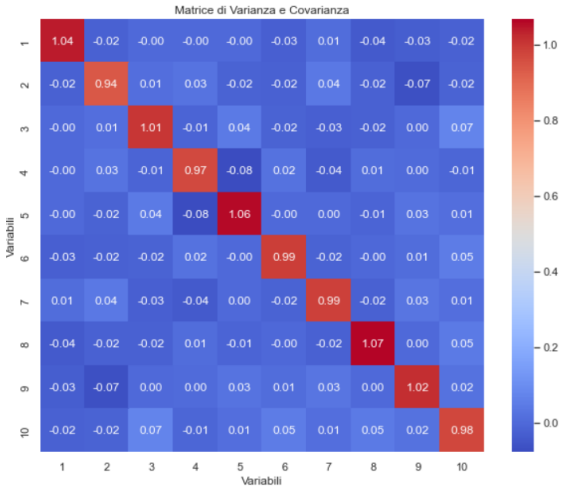


Figura 5.37: Matrice di Varianza e Covarianza del primo dataset

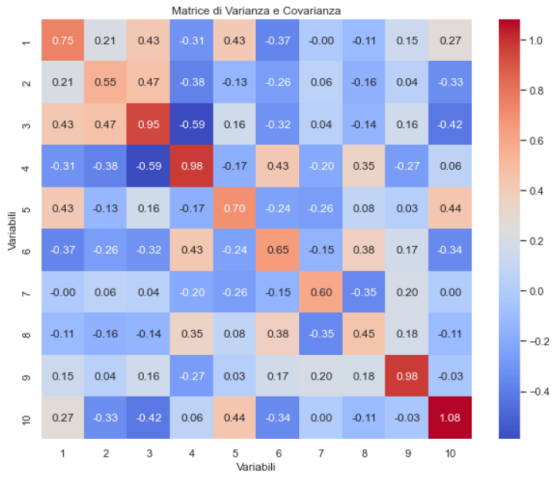
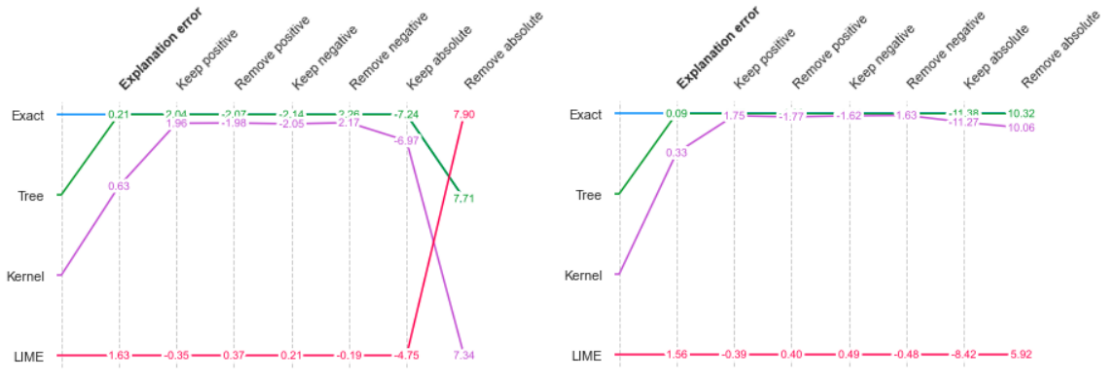


Figura 5.38: Matrice di Varianza e Covarianza del secondo dataset

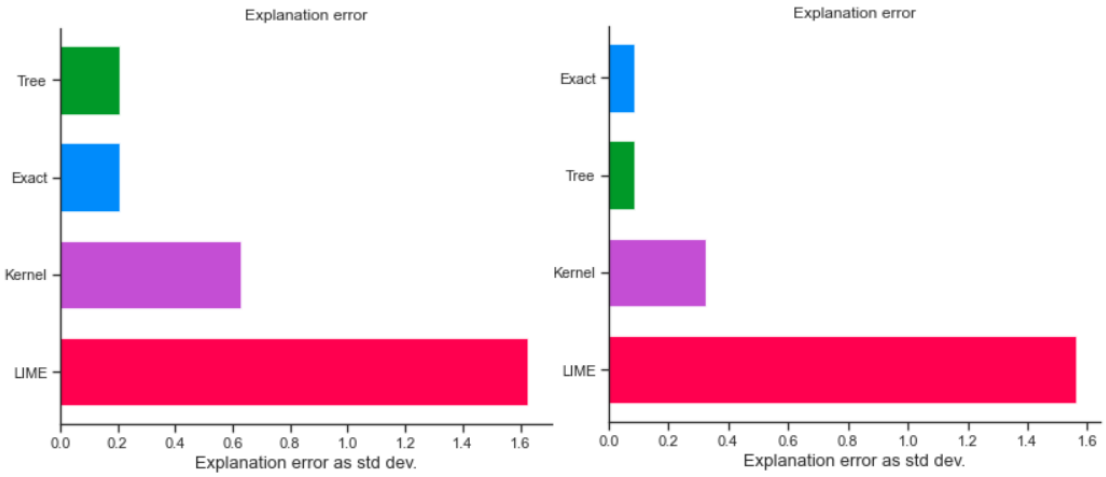
5.7.1 Risultati



(a) Valori delle metriche nel caso del primo dataset

(b) Valori delle metriche nel caso del secondo dataset

Figura 5.39: Comparazione performance dei metodi nei due dataset



(a) Explanation error nel primo dataset

(b) Explanation error nel secondo dataset

Figura 5.40: Comparazione dei metodi nei due dataset

Anche in questo contesto, i risultati ottenuti continuano a evidenziare tendenze interessanti riguardo alle performance degli algoritmi di spiegazione in relazione al grado di correlazione tra le feature dei dati.
Un notevole risultato è che l'algoritmo LIME mostra un leggero miglioramento delle performance quando le feature presentano una maggiore correlazione. Questa tendenza è in linea con quanto osservato in precedenza e suggerisce che LIME potrebbe essere una scelta appropriata per spiegare modelli di machine learning in contesti con feature più interdipendenti.

Tuttavia, è interessante notare che, in alcuni casi, anche gli algoritmi SHAP sembrano registrare un lieve miglioramento delle performance in presenza di correlazione tra le feature. Questo miglioramento, tuttavia, sembra essere dovuto a circostanze particolari, poiché gli algoritmi SHAP presuppongono l'indipendenza tra le feature. In generale, gli SHAP mostrano una tendenza a una leggera diminuzione delle performance con feature correlate, sebbene in alcune situazioni possano ancora ottenere performance paragonabili o addirittura leggermente migliori.
In sintesi, questi risultati confermano l'importanza di selezionare attentamente l'algoritmo di spiegazione in base alle specifiche caratteristiche dei dati e del modello, rafforzando l'idea che LIME sia una scelta promettente in contesti con feature più correlate, mentre gli algoritmi SHAP potrebbero richiedere un'analisi più attenta per garantire spiegazioni accurate.

Capitolo 6

Conclusioni

In conclusione, il metodo SHAP (SHapley Additive exPlanations) si è dimostrato generalmente preferibile rispetto ai metodi di spiegazione precedentemente utilizzati, come ad esempio il metodo LIME (Local Interpretable Model-agnostic Explanations). Questo verdetto è supportato da diverse ragioni che emergono dall'analisi qui condotta e dai risultati ottenuti.
Gli esperimenti condotti in questa tesi hanno una rilevanza significativa in quanto sono stati progettati per essere il più generici possibile, coprendo una vasta gamma di scenari. Si sono affrontati diversi tipi di problemi, compresi problemi di classificazione e regressione, utilizzando una varietà di algoritmi di machine learning, come XGBoost e RandomForest. I dataset utilizzati sono costituiti da dati tabulari con feature semplici, il che consente di valutare la spiegabilità delle previsioni anche da parte di un operatore umano cosa più difficile da ottenere nel caso di dataset visuali (immagini) o di testo.
Inoltre, gli esperimenti sono stati condotti in diversi contesti, compresi casi con e senza perturbazioni, e sono stati utilizzati vari tipi di metriche per analizzare in dettaglio gli effetti delle feature nell'output del modello (vedasi metriche Keep/Remove). Questi esperimenti aggiuntivi hanno esaminato l'affidabilità e la sensibilità dei metodi anche quando i prerequisiti, come l'indipendenza delle feature nel caso di SHAP, non sono stati rispettati.
Un risultato interessante emerso dagli esperimenti è che le soluzioni offerte dai metodi di SHAP si sono rivelate molto instabili in alcune circostanze, spesso meno performanti quando le feature erano fortemente correlate ed in alcuni rari casi addirittura più performanti. D'altra parte, nel caso di LIME, si è osservato un netto miglioramento delle prestazioni nella maggior parte degli esperimenti con feature più correlate (portando ad avere una conoscenza più globale anche dello stesso metodo LIME).
Uno dei punti fondamentali che ha contribuito a rendere il metodo SHAP una scelta preferita è la sua capacità di catturare in modo più completo e accurato le informazioni derivanti dalla coalizione tra le feature, fornendo così una stima più precisa dell'importanza di ciascuna di esse. Questo aspetto permette di ottenere spiegazioni delle previsioni dei modelli di machine learning più concrete e, soprattutto, più precise, riflettendo la loro correlazione intrinseca con l'output del modello.
La ragione principale alla base della superiorità di SHAP rispetto a LIME è connessa alla sua base teorica. SHAP è fondato sulla teoria matematica degli Shapley value, derivante

dalla teoria dei giochi cooperativi, fornendo così una solida fondazione per spiegare il contributo di ciascuna feature all'interno di un modello di machine learning. Diversamente LIME si basa su un approccio di perturbazione e approssimazione locale, che potrebbe non essere altrettanto stabile o accurato in tutte le situazioni.
SHAP garantisce inoltre la proprietà di "consistenza matematica", assicurando che le spiegazioni siano coerenti con le proprietà matematiche globali del modello. Questa coerenza è fondamentale per assicurare che le spiegazioni siano valide su un insieme più ampio di dati, mentre LIME potrebbe generare spiegazioni incoerenti con il modello in determinate circostanze. Un ulteriore vantaggio risiede nella sua capacità di effettuare una modellazione più precisa, prendendo in considerazione tutte le possibili combinazioni di feature durante il calcolo del contributo di ciascuna di esse alle previsioni. Questo approccio permette una stima più accurata delle interazioni tra le feature rispetto all'approccio di campionamento locale utilizzato da LIME, il quale potrebbe non catturare completamente tali interazioni complesse. SHAP si è dimostrato generalmente più stabile rispetto a LIME (nonostante in alcuni casi quest'ultimo abbia avuto una variazione di performance minore), risultando meno sensibile a piccole variazioni nei dati di input grazie alle sue rigorose fondamenta matematiche. Questa stabilità è cruciale in applicazioni in cui è necessario ottenere spiegazioni coerenti e affidabili. Infine, SHAP è compatibile con una vasta gamma di modelli di machine learning, dai modelli lineari alle reti neurali, grazie alla sua natura model-agnostic. Questa versatilità, combinata con la sua solidità teorica, rende SHAP una scelta preferita in molte situazioni in cui è essenziale ottenere spiegazioni accurate e coerenti per le previsioni dei modelli di machine learning.
Va notato che, nonostante tutti questi vantaggi, l'applicazione di SHAP potrebbe richiedere maggiori risorse computazionali rispetto a LIME, specialmente quando si utilizzano modelli di machine learning molto complessi o dataset consistenti. Tuttavia, la sua precisione, stabilità e coerenza ne fanno uno strumento potente per affrontare la sfida dell'interpretabilità nei modelli di machine learning, aprendo la strada a ulteriori sviluppi e applicazioni in questo campo in continua evoluzione.

Bibliografia

[1] *Benchmark xgboost explanations.*

[2] A. Aas, Løland, *Explaining individual predictions when features are dependent: More accurate approximations to shapley values*, arXiv preprint arXiv:1903.10464, (2019).

[3] G. Ancona, Oztireli, *Explaining deep neural networks with a polynomial time algorithm for shapley value approximation*, in Proceedings of the 36th International Conference on Machine Learning, vol. 97, PMLR, 2019.

[4] T. Castro, Gomez, *Polynomial calculation of the shapley value based on sampling*, Computers & Operations Research, 36 (2009). Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X).

[5] Y.-L. Chou, C. Moreira, P. Bruza, C. Ouyang, and J. Jorge, *Counterfactuals and causability in explainable artificial intelligence: Theory, algorithms, and applications*, Information Fusion, 81 (2021).

[6] I. Covert and S. Lee, *Improving kernelshap: Practical shapley value estimation via linear regression*, CoRR, abs/2012.01536 (2020).

[7] A. Dhurandhar, K. Shanmugam, R. Luss, et al., *Explanations based on the missing: Towards contrastive explanations with pertinent negatives*, in Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2018.

[8] European Commission, *Proposal for a regulation of the european parliament and of the council on artificial intelligence*, 2021.

[9] G. Faigle, *Least square approximations and linear values of cooperative games*, in Algebraic Techniques and Their Use in Describing and Processing Uncertainty, Springer, 2020.

[10] F.-J. Gonzalez-Diaz, Garcia-Jurado, *An Introductory Course on Mathematical Game Theory*, vol. 115 of Graduate Studies in Mathematics, 2010.

[11] L. Lundberg, *A unified approach to interpreting model predictions*, in Advances in Neural Information Processing Systems 30, B. Guyon, Luxburg, ed., Curran Associates, Inc., 2017.

[12] S. M. Lundberg, G. G. Erion, and S. Lee, *Consistent individualized feature attribution for tree ensembles*, CoRR, abs/1802.03888 (2018).

[13] C. Molnar, *Interpretable machine learning: A guide for making black box models explainable*, arXiv preprint, (2019).

[14] R. B. Myerson, *Game Theory, Analysis of Conflict*, Harvard University Press, 1991.

[15] G. Ribeiro, Singh, *Anchors: High-precision model-agnostic explanations*, in Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018.

[16] M. T. Ribeiro, S. Singh, and C. Guestrin, *"why should I trust you?": Explaining the predictions of any classifier*, CoRR, abs/1602.04938 (2016).

[17] Z. Ruiz, Valenciano, *The family of least square values for transferable utility games*, Games and Economic Behavior, 24 (1998).

[18] L. S. Shapley, *A value for n-person games*, in Contributions to the Theory of Games, volume II, T. Kuhn, ed., vol. 28 of Annals of Mathematical Studies, Princeton University Press, 1953.

[19] K. Shrikumar, Greenside, *Learning important features through propagating activation differences*, (2017).

[20] K. Strumbelj, *An efficient explanation of individual classifications using game theory*, Journal of Machine Learning Research, 11 (2010).

[21] G. Vilone and L. Longo, *Explainable artificial intelligence: a systematic review*, 2020.

[22] Weber, *Probabilistic values for games*, in Essays on the Shapley Value and Its Applications, A. E. Roth, ed., Cambridge University Press, 1988.

[23] J. Yang, *Fast treeshap: Accelerating SHAP value computation for trees*, CoRR, abs/2109.09847 (2021).