UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Laurea Triennale in Ingegneria Informatica

Progettazione e sviluppo di un wrapper per spiegare modelli machine learning per la anomaly detection in un contesto Industria 4.0

Relatore:

Ing. Antonio Luca Alfeo

Prof. Mario G.C.A. Cimino

Candidato:

Andrea Vagnoli

ANNO ACCADEMICO 2023/2024

Abstract

L'Industria 4.0 rappresenta una rivoluzione nell'ambito industriale, caratterizzata dall'uso di tecnologie digitali avanzate nei processi produttivi. L'obiettivo dell'Industria 4.0 è creare fabbriche intelligenti e connesse, in grado di operare in modo autonomo, comunicando tra di loro e adattandosi dinamicamente alle esigenze del mercato, aumentando così l'efficienza e la competitività dell'industria.

L'Intelligenza Artificiale (AI) e l'Intelligenza Artificiale spiegabile (XAI), svolgono un ruolo cruciale in questo contesto: la prima permette di ottimizzare i processi produttivi e decisionali attraverso l'impiego di algoritmi avanzati, mentre la seconda consente agli operatori del settore di comprendere il funzionamento che si cela dietro a questi sistemi intelligenti, facilitando l'adozione e l'integrazione dell'IA nell'Industria 4.0 e promuovendo una collaborazione più efficace tra umani e macchine.

Il lavoro di questa tesi si concentra in particolare sullo studio del rilevamento delle anomalie (anomaly detection): una tecnica di analisi di comportamenti anomali all'interno un processo industriale, al fine di prevedere malfunzionamenti nei macchinari industriali e garantire un'alta efficienza dell'intero ciclo produttivo.

L'obiettivo è stato analizzare un insieme di dati elaborati provenienti dal settore delle lavanderie industriali, cercando di rilevare quando un gruppo di dati nello specifico potesse essere definito "anomalo" rispetto a tutti gli altri valori. Per fare questo lavoro sono stati usati inizialmente due algoritmi di AI: MLP (Multilayer Perceptron), basato su rete neurale artificiale, ed Isolation Forest, basato invece su alberi decisionali e sviluppato in particolare per studiare il problema riguardante il rilevamento di anomalie. In seguito, tramite alcune tecniche di XAI, l'analisi si è concentrata sul capire quale delle singole grandezze del dataset avessero un'importanza relativa maggiore nel determinare un'anomalia. Nello specifico, l'impiego di alcune librerie durante questa analisi ha reso necessaria l'estensione di alcune funzionalità del modello Isolation Forest. Questo ha portato ad un ulteriore sviluppo del lavoro, con l'obiettivo di trovare una soluzione che potesse integrarsi con il modello preesistente, dovendo poi testarla valutando il suo funzionamento anche in maniera più sistematica. Grazie all'utilizzo di tecniche di XAI è stato possibile ottenere una maggiore comprensione dei dati e fornire anche alcune spiegazioni sui due modelli utilizzati, rapportando l'efficacia di ognuno di essi in funzione del problema del rilevamento delle anomalie e dei dati analizzati.

Indice

Capitolo 1

Introduzione

L'intelligenza artificiale, secondo la definizione fornita da IBM [1], è "una tecnologia che consente a computer e macchine di simulare l'intelligenza umana e le capacità di risoluzione dei problemi". Come si evince da questa definizione, l'idea alla base di questa tecnologia è quindi quella di supportare il lavoro umano nel cercare soluzione a problemi in una maniera più efficiente rispetto alle strategie più tradizionali, essendo in grado di manipolare grandi quantità di dati in poco tempo. Questa capacità consente di analizzare e comprendere i dati in modo più approfondito, rivelando relazioni altrimenti difficili da individuare. Queste relazioni possono poi essere apprese da questi sistemi in modo da diventare sempre più efficaci nella risoluzione di problemi specifici.

Per i motivi appena esposti, si può capire come l'intelligenza artificiale sia uno strumento sempre più utilizzato all'interno dei processi decisionali. Rimane tuttavia difficile comprendere il modo in cui operano questi algoritmi di AI e come avvengono nel dettaglio per svariati motivi, tra cui:

- Complessità degli algoritmi, che possono operare su molteplici livelli di astrazione rendendo difficile capire cosa influenza determinate decisioni;
- Interazioni complesse tra le variabili di input e di output che possono rendere complicato tenere traccia delle singole relazioni e che possono aumentare anche in relazione alla quantità dei dati analizzati;
- Regole non esplicite che i modelli di AI possono apprendere dai dati di addestramento e che non sono facilmente interpretabili dagli esseri umani.

In questo contesto si inserisce l'utilizzo dell'Intelligenza Artificiale spiegabile, che, come definisce nuovamente IBM [2], è "un insieme di processi e metodi che consente agli utenti umani di comprendere e fidarsi dei risultati e dell'output creati dagli algoritmi di

machine learning". Tali algoritmi altrimenti rimarrebbero con l'appellativo di "black - box", termine indicante il fatto che producono previsioni o decisioni senza fornire una spiegazione chiara su come sono state raggiunte tali conclusioni.

Riuscire a comprendere gli output prodotti da questi algoritmi permette così di effettuare scelte più consapevoli e aumentare l'affidabilità che l'uomo ha in questi algoritmi, soprattutto in relazione al contesto d'utilizzo in cui l'intelligenza artificiale spiegabile viene utilizzata, in particolare in quegli ambiti in cui le decisioni effettuate devono essere accompagnate da delle solide motivazioni, riducendo il margine di errore al minimo (sanità, finanza, sicurezza).

Lo stesso discorso può essere applicato al settore dell'Industria 4.0. Questo termine, che sta ad indicare la quarta rivoluzione sul piano industriale, introduce in maniera sistematica l'integrazione delle tecnologie digitali intelligenti (tra cui anche sistemi di intelligenza artificiale) nei processi manifatturieri e industriali. Garantire l'operabilità di un macchinario, prevedere un suo malfunzionamento o un evento anomalo che per natura risulta un fenomeno difficilmente classificabile, è senz'altro una sfida che richiede anche l'utilizzo di tecniche di XAI. Trovare una relazione tra il modello astratto e il modello fisico permette poi di andare ad operare e risolvere il problema che l'algoritmo ha predetto, ma che necessita comunque l'attribuzione di un significato contestualizzabile nel mondo reale, senza il quale gli addetti ai lavori non potrebbero andare ad operare sui dispositivi monitorati.

L'utilizzo di tecniche di XAI si rivelano pertanto essenziali anche quando sono inserite in un contesto di applicazioni industriali: il manager industriale, chiamato a prendere decisioni nella gestione e nel coordinamento delle attività produttive, non può affidarsi esclusivamente sui risultati di un algoritmo di AI, ma deve capire in maniera approfondita la logica che è dietro a tali predizioni. Solo una volta che il meccanismo che si cela dietro a questi algoritmi risulta comprensibile e chiaro agli occhi delle figure manageriali, allora possono essere prese decisioni consapevoli basate su quest'ultimi, altrimenti sarebbe difficile prendersi la responsabilità di determinate azioni senza avere delle solide basi analitiche a supporto di esse. È essenziale quindi anche in questo ambito che le analisi suggerite da questi modelli siano accompagnate da solide motivazioni che supportino il processo di decisione del manager preposto a tale compito.

È necessario quindi capire con certezza le variabili fisiche in gioco che hanno deter-

minato un rallentamento all'interno del processo produttivo, al fine di permettere agli stakeholders del settore industriale di mettere in relazione le grandezze analizzate, rendendo comprensibili dei fenomeni che altrimenti risulterebbero sconosciuti.

Capitolo 2

Related Works

Oggi, l'Industria 4.0 può essere considerata una realtà che integra tecnologie moderne e innovazioni. L'intelligenza artificiale in questo contesto può essere considerata il componente principale della trasformazione industriale che permette a macchine intelligenti di eseguire autonomamente compiti come auto-monitoraggio, interpretazione, diagnosi e analisi. Metodologie basate sull'AI supportano produttori e industrie nella previsione dei loro bisogni di manutenzione (come nel caso di rilevazione di anomalie) e nella riduzione del tempo di inattività. In questo contesto, l'intelligenza artificiale spiegabile (XAI) studia e progetta algoritmi che producono spiegazioni comprensibili dall'uomo delle informazioni e delle decisioni basate sull'AI.

Nell'articolo "From Artificial Intelligence to Explainable Artificial Intelligence in Industry 4.0: A Survey on What, How, and Where" [3], gli autori offrono una panoramica sulle applicazioni dell'intelligenza artificiale spiegabile nel contesto dell'industria 4.0, identificando le motivazioni legate allo sviluppo di questa tecnologia e le necessità dietro al suo utilizzo.

Questo articolo fornisce innanzitutto un'ampia panoramica dei metodi basati su IA e XAI applicati in diversi contesti dell'Industria 4.0. All'inizio gli autori discutono riguardo le tecnologie caratterizzanti l'Industria 4.0. Successivamente si concentrano sul presentare i diversi metodi basati su IA e XAI combinati con l'industria 4.0.

In particolare l'articolo si sofferma sull'analizzare vari domini applicativi dove viene utilizzato l'uso combinato di queste due tecnologie. Viene spiegato che l'utilizzo di modelli di IA sempre più complessi ed astratti necessita l'utilizzo di tecniche volte a spiegare ed interpretare le decisioni e gli output che questi modelli forniscono, al fine di permettere agli operatori di potersi fidare delle analisi svolte e poterle poi applicare successivamente in contesti reali con maggiore consapevolezza. Non soltanto, ma anche modelli complessi ben conosciuti dalle figure qualificate in questo campo (come AI Engineer o Data Scientist)

potrebbero dover essere spiegati ad addetti ai lavori che non possiedono le competenze per comprenderli in maniera dettagliata (come operatori della produzione industriale). Un'altra sfida che infatti permette di affrontare l'utilizzo congiunto di queste due tecniche è quella di riuscire a sviluppare modelli o algoritmi alternativi più semplici in modo da risultare comprensibili anche ai "non addetti ai lavori". Inoltre, dato che questi tipi di modelli dipendono fortemente dal tipo di dati addestrati e dal modo in cui vengono addestrati, un apprendimento errato del modello potrebbe avere conseguenze catastrofiche una volta utilizzato nei contesti applicati legati all'industria 4.0 (manutenzione predittiva, rilevamento anomalie, interazione uomo - macchina). L'utilizzo di metodi XAI permette quindi di riuscire anche a evidenziare possibili criticità di modelli risultanti da addestramenti non corretti, prevenendo le conseguenze negative legate ad un loro utilizzo senza una reale comprensione di essi.

Una delle principali tecniche utilizzate in questo ambito è quella della *feature importance*, strumento fondamentale nell'ambito dell'intelligenza artificiale spiegabile, in quanto consente di identificare quali variabili hanno il maggiore impatto sulle decisioni di un modello di machine learning. Questo concetto è rimarcato dall'articolo "Comparison of feature importance measures as explanations for classification models" [4], dove viene affermato precisamente: "le spiegazioni più comuni per i modelli di classificazione sono le importanze delle feature". In particolare, nell'articolo citato vengono messe a confronto varie tecniche di feature importance per poter valutare le loro performance sia in maniera singolare che in relazione a tecniche diverse. Il dominio applicativo preso in esame nel paper è il settore medico, dove, citando nuovamente l'articolo, "non è solo di grande importanza prevedere l'esito clinico di un paziente, ma anche considerare le caratteristiche di questo paziente (ad esempio, età, uso di farmaci etc.) in modo spiegabile e quantificabile e quanto queste singole caratteristiche influenzino nel complesso l'esito clinico finale". Al di là del contesto preso in esame, che è differente da quello analizzato in questa tesi, è importante citare le conclusioni finali raggiunte nell'articolo: "una combinazione di diverse tecniche di feature importance potrebbe fornire risultati più affidabili e degni di fiducia". Il lavoro si conclude sottolineando il concetto che le diverse tecniche di feature importance offrono vari vantaggi a seconda del contesto e dell'applicazione, suggerendo l'uso combinato di più metodi per ottenere spiegazioni più affidabili e complete.

Diventa quindi un passaggio fondamentale utilizzare queste tecniche di valutazione

di importanza di ogni singola caratteristica del dataset oggetto di studio per dare una spiegazione sensata alle previsioni del modello di analisi.

In questo contesto si inserisce anche il concetto legato al rilevamento di anomalie, che come viene riportato nell'articolo "Explainable artificial intelligence (XAI) enabled anomaly detection and fault classification of an industrial asset" [5] viene spiegato e introdotto nel seguente modo:

"La capacità degli algoritmi di rilevamento delle anomalie di individuare la prima indicazione del guasto di un asset (macchinario) è vitale per le soluzioni successive di manutenzione predittiva, poiché consente la pianificazione delle azioni di manutenzione prima che l'asset subisca danni significativi. Il rilevamento delle anomalie è una competenza molto significativa che riduce i tempi di fermo non pianificati e la manutenzione non necessaria, consentendo una gestione più efficace dei componenti critici."

Emerge quindi come l'utilizzo di queste tecniche combinate, anche nel dominio applicativo del settore industriale, permetta effettivamente un guadagno in termini di produttività delle attività umane, abbassamento dei costi ed impiego delle risorse.

Il lavoro di questa tesi si concentra in particolare sul confronto di alcuni modelli di anomaly detection addestrati su dati relativi ad un contesto manifatturiero di Industria 4.0. Lo scopo dell'analisi è quello di riuscire a confrontare modelli diversi tramite tecniche XAI di feature importance al fine di comprendere i processi decisionali dei diversi modelli utilizzati per classificare i vari dati presi in esame.

Capitolo 3

Design e Implementazione

In questa sezione verranno spiegati i casi d'uso del sistema, la descrizione completa di tutti gli strumenti utilizzati al fine di procedere con l'analisi dei dati della tesi ed, infine, verrà illustrata l'implementazione software utilizzata per procedere con il lavoro descritto.

3.1 Use Case

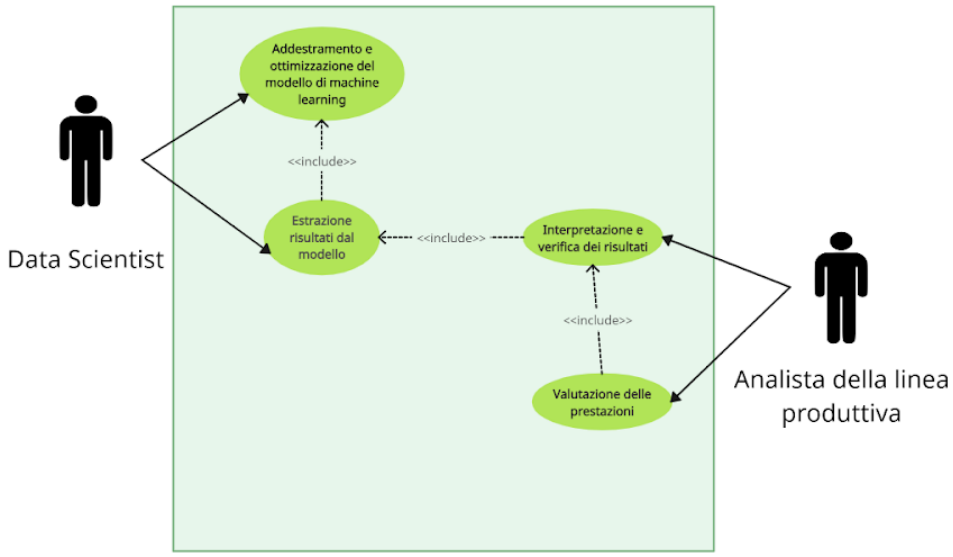


Figure 1: Use Case

Nella figura è rappresentato un possibile caso d'uso:

Attori:

- **Data Scientist**

  Attore secondario che si occupa dello sviluppo del modello. Questa figura si mantiene costantemente in contatto con l'operatore della produzione per ricevere feedback sulle prestazioni del modello e, in caso, effettuare modifiche su di esso al fine di aumentare la performance.

Azioni:

- **Addestramento e ottimizzazione del modello di machine learning**
  Questo è il compito principale di questa figura. Scelto l'asset o l'insieme di asset da monitorare il Data Scientist si occupa di sviluppare un modello opportuno che meglio si adatti al caso specifico. In seguito si occupa anche della fase di addestramento del modello per ottimizzarne al massimo l'accuratezza.
- **Estrazione di risultati dal modello**
  Una volta che il modello è addestrato, il Data Scientist si deve anche occupare di rendere comprensibili i risultati astratti del modello all'operatore della produzione, che non avrebbe altrimenti le conoscenze per trarre conclusioni pratiche.

- **Analista della linea produttiva**
  L'operatore della produzione è colui che si occupa del ciclo vita dell'asset o del gruppo di asset che vengono monitorati grazie al modello implementato. Il suo compito è anche quello di fornire feedback al Data Scientist per modificare eventualmente il modello implementato.
  Azioni:
  - **Interpreta e verifica i risultati ottenuti**
    L'operatore della produzione si occupa di verificare se la predizione suggerita dal modello risulti reale, nel nostro caso di studio questa figura deve verificare in particolare se ad esempio un asset monitorato identificato dal modello come "soggetto ad anomalia" si sia realmente guastato o presenti un problema di funzionamento.
  - **Valuta le prestazioni**
    L'operatore, una volta verificata la predizione del modello, deve anche valutare se il modello ha realmente indicato un risultato corretto ed in seguito informare il Data Scientist che dovrà eventualmente effettuare delle modifiche.

3.2 Design

L'analisi è partita innanzitutto dall'utilizzo di due modelli di addestramento differenti sul dataset di studio.

3.2.1 Modello di classificazione

Un modello di classificazione è un tipo di modello utilizzato nell'ambito del machine learning per predire l'appartenenza di un'istanza (un insieme di dati) a una o più categorie predefinite. In altre parole, un modello di classificazione prende in input un insieme di caratteristiche e produce una classe (un valore numerico) che rappresenta la categoria a cui quella specifica istanza appartiene. Possiamo valutare la precisione del modello in vari modi, tra cui lo score di accuratezza, che rappresenta la percentuale di quanti dati di test riesce a classificare correttamente rispetto al totale delle etichette di test.

3.2.2 Anomaly detection

L'anomaly detection, o rilevamento delle anomalie, è una tecnica utilizzata nell'ambito del machine learning e dell'analisi dei dati per identificare istanze di dati che si discostano significativamente dalla maggioranza del resto dei dati presi in esame. Nel nostro caso specifico, le istanze di dati considerate anomale vengono classificate nella classe 1, mentre le istanze di dati che risultano essere non anomale vengono classificate nella classe 0.

3.2.3 MLPClassifier

Il primo modello utilizzato è stato MLPCLassifier (Multi-layer Perceptron Classifier). Il MLPClassifier è un tipo di modello di classificazione che utilizza una rete neurale artificiale chiamata Multi-Layer Perceptron (MLP). Questa rete è composta da tre tipi di strati di neuroni: strato di input, che riceve i dati di input, strati nascosti, che sono uno o più strati intermedi tra lo strato di input e lo strato di output. Ogni strato nascosto contiene neuroni che trasformano le informazioni di input in modi complessi e non lineari, catturando così i pattern nei dati. Lo strato di output invece produce le previsioni finali del modello. Può contenere un singolo neurone per problemi di classificazione binaria o

più neuroni per problemi di classificazione multiclasse, ognuno dei quali rappresenta una classe diversa.

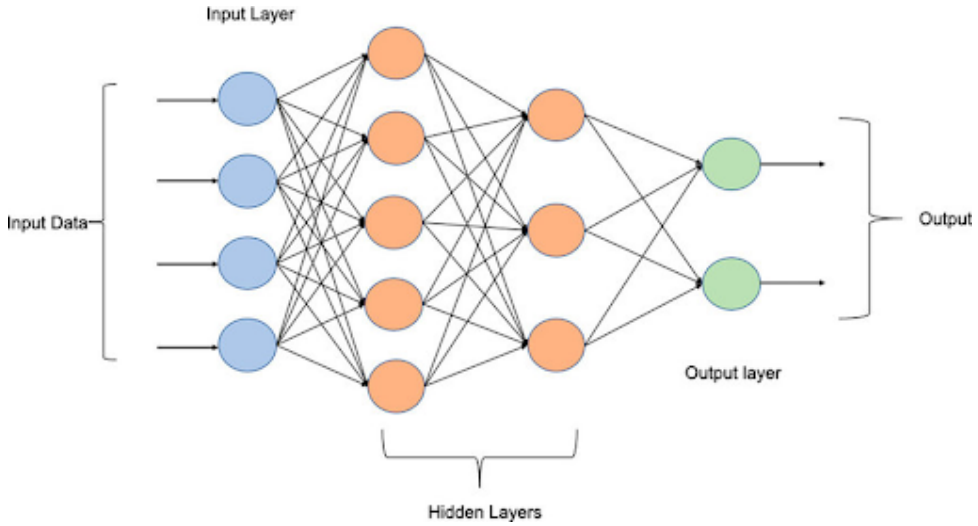


Figure 2: Principio di funzionamento di MLPClassifier

Gli strati nascosti sono chiamati così perché i loro neuroni non sono direttamente collegati agli input o agli output del modello e quindi sono "nascosti" all'esterno. Questi strati sono responsabili della maggior parte del lavoro di elaborazione dei dati all'interno del modello, catturando le relazioni complesse tra le caratteristiche di input e le classi di output. Il classificatore MLP utilizza quindi questi strati nascosti per apprendere e rappresentare i pattern nei dati di addestramento, che può quindi utilizzare per fare previsioni accurate su nuovi dati non visti.

Gli **iperparametri** di un MLP (Multi-Layer Perceptron) sono i parametri del modello che non vengono appresi durante il processo di addestramento, ma che devono essere impostati prima dell'avvio dell'addestramento stesso. Questi iperparametri influenzano il comportamento e le prestazioni complessive del modello. Alcuni dei parametri che si possono scegliere sono:

- **hidden_layer_sizes**: un array che rappresenta il numero di neuroni per ogni strato nascosto della rete neurale;
- **activation**: la funzione di attivazione utilizzata negli strati nascosti della rete neurale. Può essere 'identity', 'logistic', 'tanh' o 'relu';
- **solver**: il metodo utilizzato per l'ottimizzazione dei pesi della rete neurale. Può essere 'lbfgs', 'sgd' o 'adam';

- **learning_rate**: il metodo per aggiornare i pesi durante l'addestramento. Può essere 'constant', 'invscaling' o 'adaptive';
- **max_iter**: il numero massimo di iterazioni durante l'addestramento.

3.2.4 Isolation Forest

Il secondo modello utilizzato è stato il modello Isolation Forest. Quest'ultimo è un algoritmo di machine learning per il rilevamento specifico delle anomalie. Come suggerisce il nome, utilizza la media delle previsioni di diversi alberi decisionali assegnando tale punteggio finale di anomalia ad un certo punto dei dati. A differenza di altri algoritmi di rilevamento delle anomalie, che prima definiscono cosa sia "normale" e poi segnalano qualsiasi altra cosa come anomala, Isolation Forest cerca di isolare i punti dati anomali fin dall'inizio. Il punteggio di anomalia è inversamente associato alla lunghezza del percorso poiché segue l'idea di base che le anomalie, per definizione delle stesse, necessitano di meno suddivisioni per essere isolate, dato che sono poche e diverse rispetto alla maggioranza dei dati.

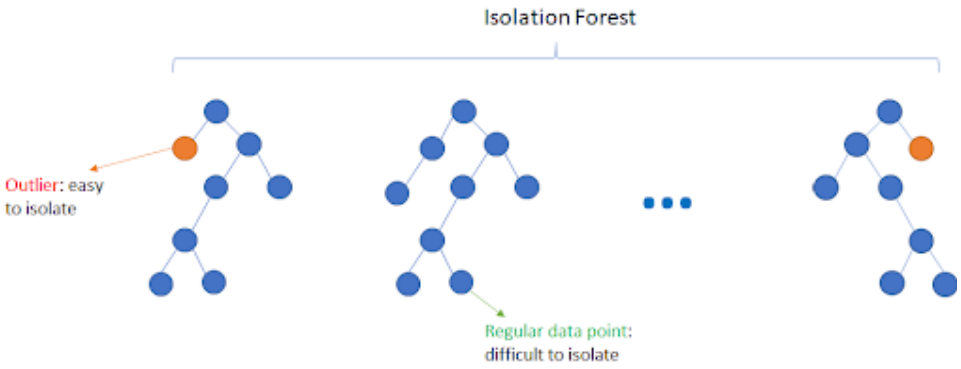


Figure 3: Principio di funzionamento di Isolation Forest

Successivamente all'addestramento dei due modelli con il dataset di studio e l'osservazione dei due diversi punteggi di accuratezza, sono state usate tecniche di XAI per cercare di spiegare tali previsioni.

3.2.5 Feature Importance

L'importanza di una feature misura quanto un determinato valore di una caratteristica influenza l'output del modello. In questo lavoro sono state utilizzate alcune tecniche di

feature importance, come *SHAP* e *Permutation Importance*, che vengono riportate qui di seguito.

3.2.6 SHAP

I valori di Shapley sono una tecnica di attribuzione dell'importanza delle caratteristiche utilizzata nell'ambito del machine learning, che quantificano il contributo di ciascuna caratteristica nell'output di un modello predittivo, considerando tutte le possibili combinazioni di caratteristiche e valutando quanto ciascuna caratteristica contribuisca in media a migliorare la previsione quando combinata con le altre. In particolare, questo indice proviene dalla teoria dei giochi, e viene utilizzato per distribuire equamente i guadagni o i costi tra i partecipanti di un gioco cooperativo, in base al loro contributo al guadagno totale.

Formulazione Matematica

In SHAP, una predizione $f(x)$ di un modello f è scomposta in una somma lineare di contributi di ogni caratteristica xi, rispetto ad un valore di base $E[f(x)]$:

$$f(x) = E[f(x)] + \sum_{i=1}^{M} \phi_i \cdot x_i$$

dove:

- $f(x)$ è la previsione del modello per il dato x.
- $E[f(x)]$ è la previsione media del modello.
- ϕ_i è il valore SHAP per la caratteristica i.
- x_i è il valore della caratteristica i.
- M è il numero totale delle caratteristiche.

Il calcolo del valore SHAP ϕ_i per la caratteristica i è dato da:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \left[f(S \cup \{i\}) - f(S) \right]$$

dove:

- S è un sottoinsieme delle caratteristiche N che non include i.
- $|S|$ è la dimensione del sottoinsieme S.
- $|N|$ è il numero totale delle caratteristiche.
- $f(S)$ è la funzione di valutazione del modello per il sottoinsieme S.
- $f(S \cup \{i\})$ è la funzione di valutazione del modello quando la caratteristica i è aggiunta al sottoinsieme S.

Anche in questo caso, i valori di Shapley possono essere visualizzati tramite grafici che permettono una comprensione più intuitiva del contributo delle singole features.

3.2.7 Permutation Importance

La permutation importance è una tecnica utilizzata nell'ambito del machine learning per valutare l'importanza relativa delle variabili di input in un modello predittivo. La procedura consiste nel calcolare l'importanza di ciascuna variabile di input valutando quanto le prestazioni del modello peggiorano quando le osservazioni di una particolare variabile vengono permutate rispetto a quando le osservazioni sono ordinate in modo corretto. Se la permutazione casuale di una variabile non influisce significativamente sulle prestazioni del modello, allora si può dedurre che quella variabile non sia molto importante per il modello. Al contrario, se la permutazione di una variabile causa un significativo peggioramento delle prestazioni del modello, allora si può concludere che quella variabile è importante per il modello.

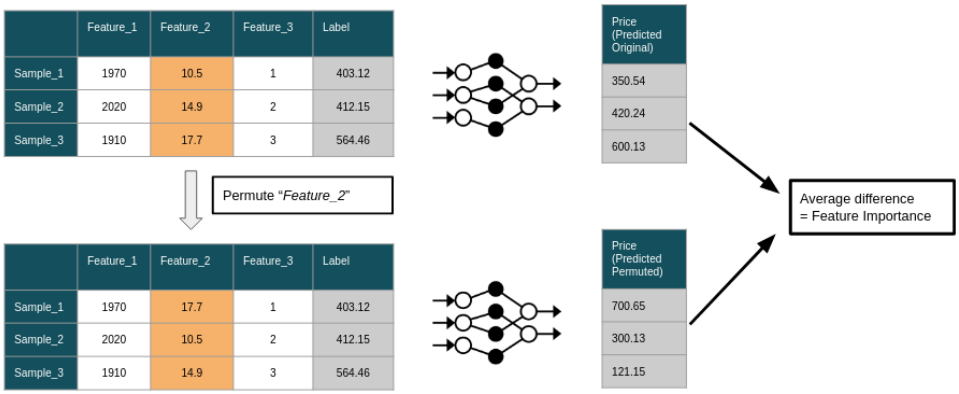


Figure 4: Principio di funzionamento di Permutation Importance

3.2.8 Counterfactuals

I controfattuali (counterfactuals) analizzano le variazioni dell'output di un modello in seguito a variazioni dei valori di input. L'utilizzo dei controfattuali fornisce una risposta alla domanda "quali dati e come dovrei cambiarli per fare in modo che il modello li classifichi in un'etichetta differente rispetto a quella di partenza?". I modelli di machine learning possono quindi essere utilizzati per generare controfattuali che ci consentono di comprendere come il cambiamento di determinate variabili o condizioni potrebbe influenzare i risultati.

Formulazione Matematica

Supponiamo di avere un modello di classificazione $f()$ che mappa l'input x in un output y. Il problema della generazione di un controfattuale può essere formalmente espresso come segue:

$$f(x) = y$$

Vogliamo trovare un controfattuale x' tale che:

$$f(x') = y' \text{ con } y' \neq y$$

Inoltre, x' dovrebbe essere "vicino" a x per minimizzare le differenze non rilevanti. Questo può essere formalizzato come un problema di ottimizzazione:

$$\min_{x'} \text{ d(x,x')}$$

dove $d(x, x')$ è una funzione di distanza che misura la differenza tra x e x'.

Formule e Algoritmi per la Generazione di Controfattuali

Uno degli algoritmi più noti per la generazione di spiegazioni controfattuali è DiCE (Diverse Counterfactual Explanations). L'algoritmo cerca di generare una serie di controfattuali che siano diversi tra loro e vicini agli input originali.
L'ottimizzazione può essere espressa come:

$$\min_{x'} \lambda d(x, x') + 1\{f(x') \neq y\}$$

dove:

- $d(x, x)$ è una funzione di distanza, ad esempio la distanza euclidea.
- $1\{f(x') \neq y\}$ è un indicatore che vale 1 se f(x') non è uguale a y e 0 altrimenti.
- λ è un parametro di bilanciamento.

I counterfactuals possono essere visualizzati tramite grafici che verranno illustrati in seguito e che che permettono una comprensione più intuitiva del contributo delle singole features.

3.3 Implementazione

Il software è stato implementato in Python con l'utilizzo di alcune librerie: *Sklearn*, *Matplotlib*, *Numpy*, *Pandas*, per l'utilizzo dei modelli e il calcolo delle principali metriche utilizzate e la manipolazione del dataset e le librerie *SHAP*, *DiCE*, *CFNOW* per il calcolo delle metriche relative alle tecniche di intelligenza artificiale spiegabile.

3.3.1 Approfondimento Wrapper modello Isolation Forest

Come riportato precedentemente, alcune librerie di XAI utilizzate durante l'analisi richiedevano dei metodi che il modello Isolation Forest della libreria Sklearn non presentava inizialmente. Il calcolo delle feature importance con le varie librerie elencate richiede in particolare due metodi di cui il modello Isolation Forest era inizialmente sprovvisto:

- **predict_proba**, una funzione che permetta di calcolare la probabilità di appartenenza alle due classi analizzate (la classe 0 e la classe 1) di un determinato input fornito in ingresso alla funzione;
- **score**, una funzione che, date in ingresso le feature e i dati di test, permette di calcolare lo score di accuratezza complessivo del modello.

Per poter continuare l'analisi è stato deciso di implementare questi due metodi mancanti, costruendo un wrapper del modello Isolation Forest.

`class CustomIsolationForest(IsolationForest, distance_min, distance_max)`

parametri:

- **IsolationForest**: classe di partenza che viene estesa dalla classe CustomIsolation-Forest. Il costruttore di CustomIsolationForest inizializza anche tutti gli attributi già appartenenti classe IsolationForest (omesse in questa documentazione);
- **distance_min**: attributo che viene inizialmente inizializzato a None, rappresenta un parametro che verrà utilizzato nei metodi init_predict_proba e predict_proba;
- **distance_max**: attributo che viene inizialmente inizializzato a None, rappresenta un parametro che verrà utilizzato nei metodi init_predict_proba e predict_proba.

`init_predict_proba(values)`

Per ogni valore di values, calcola il metodo decision_function della classe IsolationForest. Successivamente, assegna ai due attributi distance_max e distance_min il valore massimo ed il valore minimo dei valori calcolati.
parametri: values, valori che rappresentano le variabili indipendenti del mio dataset.

`predict_proba(value)`

Tramite una funzione di normalizzazione, calcola la probabilità che la feature value appartenga alla classe 0 o alla classe 1.
parametri: value, rappresenta una singola feature.
restituisce: un vettore di due elementi, il primo elemento rappresenta la probabilità che la feature appartenga alla classe 0, il secondo elemento che appartenga alla classe 1.
La funzione di normalizzazione usata è stata la seguente:

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

`score(x_test, y_test)`

Per ogni valore di x_test, calcola la funzione predict della classe IsolationForest, che restituisce il valore 1 o -1 a seconda che la feature venga identificata come anomalia o non anomalia. A questo punto il valore ottenuto tramite la funzione predict viene mappato

nei valori 0 ed 1 (1 viene mappato in 0 e -1 in 1). Infine viene applicata la funzione accuracy_score prendendo in input il vettore risultante dalle operazioni precedenti e y_test.
parametri:

- x_test, rappresenta la parte dei dati di test del modello delle variabili indipendenti;
- y_test, rappresenta la parte dei dati di test del modello delle variabili dipendenti.

restituisce: l'output della funzione accuracy_score.
L'implementazione di tale wrapper, chiamato *CustomIsolationForest* ha permesso di aggirare il problema presentato all'inizio e quindi di continuare l'analisi ed il confronto con il modello MLPClassifier. Successivamente a tale lavoro, si è deciso di testare in maniera più approfondita il wrapper costruito, per capire se tale implementazione fosse compatibile anche con altre librerie di XAI e se i risultati ottenuti con librerie diverse potessero risultare compatibili, indicando quindi una prova a favore dell'implementazione del wrapper del modello.

3.3.2 Dataset

Al fine di gestire al meglio i dati è stata implementata anche una classe Dataset.

```
class Dataset(dataset_name, target_feature_name)
```

parametri:

- **dataset_name**: nome del dataset analizzato;
- **target_feature_name**: nome della feature target del dataset.

Il costruttore della classe istanzia le seguenti variabili della classe:

- **dataset_name**: nome del dataset;
- **target**: nome della feature target del dataset;
- **x_train**: parte del dataset riservato all'addestramento del modello;
- **x_test**: parte del dataset riservato al test del modello;
- **y_train**: parte delle feature riservato all'addestramento del modello;

- **y_test**: parte delle feature riservato al test del modello.

`get_dataset_train()`

Restituisce il dataset (variabili di input e variabile di target) relativo alla parte di addestramento del modello.

`get_dataset_test()`

Restituisce il dataset (variabili di input e variabile di target) relativo alla parte di test del modello.

`get_X()`

Restituisce il dataset completo privo delle feature target del modello.

`get_Y()`

Restituisce la colonna delle variabile target del dataset di studio.

`get_shapley_values(model)`

Restituisce i valori Shapley relativi al dataset e calcolati col modello dato in ingresso.
parametri: model, modello addestrato sui dati del dataset.

`get_greedy_chart(model, row)`

Restituisce i diagrammi greedy_chart relativi al dataset e calcolati col modello dato in ingresso estraendo i dati da una particolare riga del dataset.
parametri:

- model, modello addestrato sui dati del dataset;
- row, riga del dataset da cui si vuole estrarre i dati per calcolare il diagramma.

`get_countershap(model, row)`

Restituisce i diagrammi CounterShapley relativi al dataset e calcolati col modello dato in ingresso estraendo i dati da una particolare riga del dataset.
parametri:

- model, modello addestrato sui dati del dataset;
- row, riga del dataset da cui si vuole estrarre i dati per calcolare il diagramma.

Capitolo 4

Case Study

Il caso di studio di questa tesi riguarda l'analisi di un dataset contenente informazioni sul consumo di energia di asset relativi ad un'azienda di lavanderia industriale.

In particolare, gli asset analizzati sono stati 3: una lavatrice industriale in grado di operare più funzioni (prelavaggio, lavaggio etc.), un macchinario che svolge la stessa funzione del ferro da stiro in ambito domestico e una lavatrice progettata in particolare per la fase aspirante e di centrifuga.

Di questi tre asset sono stati analizzati i consumi di energia attraverso tre serie temporali con una misurazione ogni secondo (86400 misurazioni al giorno).

L'analisi si è concentrata sui consumi di energia in quanto ogni macchinario elencato era già fornito di un controllore integrato in grado di effettuare tale misurazioni. Altre grandezze utili a questa analisi, come le emissioni acustiche o l'analisi delle vibrazioni dei tre componenti avrebbero richiesto l'ausilio di ulteriori sensori non presenti di default negli asset.

Dalle serie temporali si è deciso di classificare come "anomali", determinati valori per i quali il consumo di energia variava in maniera non trascurabile rispetto al valore in condizione di regime.

Infine, sono stati aggiunti a questi dati anche altre misurazioni, relative a serie temporali nelle quali si sono quantificate le vibrazioni di cuscinetti industriali, al fine di validare l'analisi di anomaly detection attraverso un contesto di dati più generalizzato. Il dataset finale riassume quindi le informazioni relative a 3 asset di lavanderia industriale (3 serie temporali, PC1 - PC2 - PC3) e a dei cuscinetti industriali (5 serie temporali BV1 - BV2 - BV3 - BV4 - BV5).

4.1 Descrizione serie temporali di origine del dataset

Vengono adesso riportate due tabelle, la prima si riferisce in particolare ai tre asset monitorati, riportando per ognuno di essi alcuni range di valori che stanno a indicare:

- kWh Operation, il consumo di energia "a regime" dell'asset;
- kWh Downtime, il consumo di energia durante le fasi di stop o di basso lavoro;
- Short Downtimes per day, numero di occorrenze al giorno in cui si presentavano le fasi di stop o di rallentamento del lavoro di breve durata;
- Downtimes per day, numero di occorrenze al giorno in cui si presentavano le fasi di stop o di rallentamento del lavoro di una durata più lunga rispetto alla precedente;
- Downtimes duration [min], intervallo di minuti dei rallentamenti di lunga durata.

La seconda tabella invece, descrive l'intervallo di campionamento, la frequenza di campionamento e il numero di dati di train e test con relative occorrenze di anomalie in percentuale riferite sia alle serie temporali degli asset (PC1 - PC2 - PC3) che alle serie temporali dei cuscinetti (BV1 - BV2 - BV3 - BV4 - BV5).

Asset	**kWh Op.**	**kWh Dt.**	**Sh.Dt. per day**	**Dt. per day**	**Dt. Dur. [min]**
CBW	[2, 4]	[0.25, 0.75]	[2, 4]	[1, 1]	[60, 240]
Washer-extractor	[3.5, 4.6]	[0.5, 3]	[3, 5]	[0, 1]	[15, 15]
Ironer	[15, 20]	[8, 12]	[1, 3]	[2, 3]	[45, 180]

Table 1: Descrizione caratteristiche per ogni asset

Name	**TS Dur.**	**Sampl. Rate**	**Sampl. per TS**	**# Inst.**	**# Train**	**# Test**	**% Anomal.**
PC1 - PC3	24 hours	1 Hz	86400	1000	665	300	5%
BV1 - BV5	40 seconds	25 Hz	1000	600	400	180	5%

Table 2: Descrizione delle serie temporali

4.2 Dataset

Il dataset è composto da 44 righe e 15 features, ed è ottenuto tramite il calcolo di alcuni indici statistici a partire dalle serie temporali analizzate precedentemente. La feature anomalous rappresenta l'etichetta e può assumere come valore 0 (rappresentato da 40 esempi su 44 righe) oppure 1 (rappresentato da 4 esempi su 44 righe), che indicano

rispettivamente l'assenza o il rilevamento di anomalia. Il dataset contiene i seguenti tipi di informazioni, rappresentando ogni informazione come una colonna:

- 90°, 75°, 50°, 25° percentile delle serie temporali;
- (MAD) Deviazione Media Assoluta delle serie temporali;
- Skewness delle serie temporali: Asimmetria delle serie temporali;
- Numero di intervalli temporali continui con valori superiori al percentile al 90°, 75°, 50°, 25° delle serie temporali;
- Numero di campioni superiori al 50% e al 25% del massimo giornaliero delle serie temporali;
- La differenza tra la deviazione media assoluta della finestra temporale corrente e la deviazione media assoluta dell'intera serie temporale.

Capitolo 5

Setup Sperimentale

5.1 Metriche Utilizzate

5.1.1 Accuracy Score

L'accuracy score, è una misura utilizzata per valutare le prestazioni di un modello di classificazione nell'ambito del machine learning. Questa misura fornisce una valutazione della capacità del modello di classificare correttamente le istanze dei dati nell'etichetta opportuna. L'accuracy score indica la percentuale di previsioni corrette rispetto al numero totale di previsioni effettuate dal modello. Viene calcolato come:

$$\text{Accuracy score} = \frac{\text{Numero di previsioni corrette}}{\text{Numero totale di previsioni}}$$

5.1.2 Spearman Correlation

La correlazione di Spearman è una misura di correlazione statistica utilizzata per valutare il grado di associazione tra due variabili. Tale correlazione valuta la relazione monotona tra le variabili. Questo significa che può catturare relazioni non necessariamente lineari tra le variabili. Il coefficiente di correlazione di Spearman è calcolato con la seguente formula:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

Dove:

- ρ è il coefficiente di correlazione di Spearman;
- d_i sono le differenze tra i ranghi delle coppie di valori;
- n è il numero totale di coppie di valori.

Il rango viene attribuito ai due gruppi di variabili (ipoteticamente y e x), ordinando in maniera crescente i due sottogruppi e assegnando un intero positivo crescente ad ogni variabile. Una volta fatto ciò, possono essere calcolate le differenze (e quindi anche il quadrato delle differenze) tra ranghi di due variabili y e x corrispondenti.
Il coefficiente di correlazione di Spearman può variare tra -1 e 1, dove:

- 1 indica una correlazione perfettamente monotona crescente;
- -1 indica una correlazione perfettamente monotona decrescente;
- 0 indica nessuna correlazione monotona tra le variabili.

5.1.3 p-value

Il p-value, nei test per verificare le ipotesi, rappresenta la probabilità che i risultati osservati durante il test siano altrettanto o meno compatibili con l'ipotesi nulla, ossia l'ipotesi ipotizzata come vera prima di effettuare il test. Durante un test di ipotesi, si stabilisce un'ipotesi nulla e un valore di soglia (solitamente 0,05) che rappresenta il livello di significatività del test. Se il p-value calcolato dai dati osservati è maggiore di del livello di soglia, allora l'evidenza empirica non contraddice sufficientemente l'ipotesi nulla e quindi non viene rifiutata. Se il p-value è minore o uguale al valore di soglia, l'evidenza empirica contraddice fortemente l'ipotesi nulla e quindi viene rifiutata, indicando che i dati osservati non sono statisticamente significativi. Tuttavia, se il p-value è approssimativamente uguale al valore di soglia, è necessario prestare attenzione e contestualizzare il risultato in base al tipo di dati analizzati e alla natura dell'analisi effettuata.
Per calcolare il p-value, utilizziamo specifiche formule che variano a seconda del test statistico in questione e della distribuzione delle variabili. Di seguito, viene presentata la formula generale per calcolare il p-value nel caso di utilizzo con la correlazione di Spearman. Nel caso della correlazione di Spearman, il p-value è usato per valutare l'ipotesi nulla che non esista una correlazione monotona tra le due variabili. Supponiamo di avere il coefficiente di correlazione di Spearman ρ calcolato per due variabili e vogliamo determinare se questo valore è statisticamente significativo.

Formula per calcolare il p-value

Il calcolo esatto del p-value dipende dalla *distribuzione t di Student* per la statistica del

test, che si basa sul coefficiente di correlazione di Spearman. Il test t con correlazione di Spearman è utilizzato per valutare la significatività statistica di un coefficiente di correlazione di Spearman tra due variabili. La statistica t è calcolata utilizzando la seguente formula:

$$t = \frac{\rho\sqrt{n-2}}{\sqrt{1-\rho^2}}$$

Dove:

- t è la statistica t di Student;
- ρ è il coefficiente di correlazione di Spearman;
- n è il numero di coppie di dati.

Questa statistica t è utilizzata per determinare se il coefficiente di correlazione di Spearman osservato è significativamente diverso da zero, indicando una correlazione tra le variabili.

Il p-value è calcolato utilizzando la distribuzione t di Student con $n-2$ gradi di libertà. La formula generale è:

$$p = 2 \cdot (1 - T_{\text{df}}(t))$$

Dove:

- $T_{\text{df}}(t)$ è la funzione di distribuzione cumulativa (CDF) della distribuzione t di Student con $\text{df} = n - 2$ gradi di libertà;
- t è la statistica del test calcolata.

5.2 Grafici utilizzati

5.2.1 CounterShapley

Grafico a linee presente all'interno della libreria CFNOW, che permette di osservare, le variazioni eseguite alle singole feature in ordine di importanza in una singola istanza di dati all'inizio classificata con una probabilità maggiore nella classe 0 (non anomalia), in modo che, una volta effettuate le modifiche, l'istanza di dati venga classificata nella classe

1 con una più alta probabilità. La scelta viene effettuata considerando tutte le possibili combinazioni.

5.2.2 Greedy chart

Grafico presente all'interno della libreria CFNOW, che permette di osservare, attraverso un cammino greedy, le variazioni eseguite alle singole feature, in ordine di importanza, in una singola istanza di dati all'inizio classificata con una probabilità maggiore nella classe 0 (non anomalia), in modo che, una volta effettuate le modifiche, l'istanza di dati venga classificata nella classe 1 con una più alta probabilità. La scelta viene effettuata in maniera greedy, considerando ogni volta la feature che aumenta maggiormente lo score.

5.2.3 Shapley values

Grafico presente all'interno della libreria SHAP che permette di osservare in ordine di importanza i valori Shapley calcolati. In particolare per ogni feature è possibile osservare il "contributo" che fornisce alla classe 0 e alla classe 1.

5.2.4 Confusion Matrix

Una matrice di confusione rappresenta in modo tabulare i risultati delle predizioni effettuate da un modello confrontandole con le etichette corrette. Ogni riga della matrice rappresenta le istanze di una classe reale, mentre ogni colonna rappresenta le istanze di una classe predetta.

Capitolo 6

Risultati Sperimentali

6.1 Analisi modelli MLPClassifier e CustomIsolationForest

L'analisi è partita innanzitutto addestrando un classificatore MLP con i dati oggetto dello studio, utilizzando una ricerca a griglia *(GridSearchCV)* per identificare i migliori parametri che permettessero di avere il grado di accuratezza maggiore. In seguito è stata usato il metodo *predict* del *MLPClassifier* per fare delle previsioni sui dati di test. Queste previsioni sono state usate attraverso la funzione *accuracy_score* per valutare l'efficacia del modello appena addestrato. In particolare i parametri testati e i valori ottenuti attraverso la ricerca a griglia sono i seguenti:

- activation: "tanh";
- "hidden_layer_sizes": [10, 30, 10];
- "learning_rate": "constant";
- "max_iter": 200;
- "solver": "adam".

Il modello MLPClassifier addestrato con gli iperparametri trovati forniva un'accuratezza del **88,889%**.
In secondo luogo l'analisi è stata ripetuta sul dataset ma questa volta tramite il modello *CustomIsolationForest*. In particolare, in maniera analoga con l'altro classificatore, una volta addestrato il modello è stata usato il metodo score (metodo del wrapper del modello) per calcolare l'accuratezza delle predizioni. Il modello CustomIsolationForest riportava in questo caso un'accuratezza del **94,444%**.

6.2 Analisi sui grafici CounterShapley e greedy chart

Il passo successivo dell'analisi è stato quello di utilizzare tecniche di intelligenza artificiale spiegabile per mostrare le predizioni ottenute dai due modelli. Sono stati calcolati, per ogni input e per ogni modello, i grafici *CounterShapley* e *greedy chart* come è illustrato nella figura sotto a titolo di esempio.

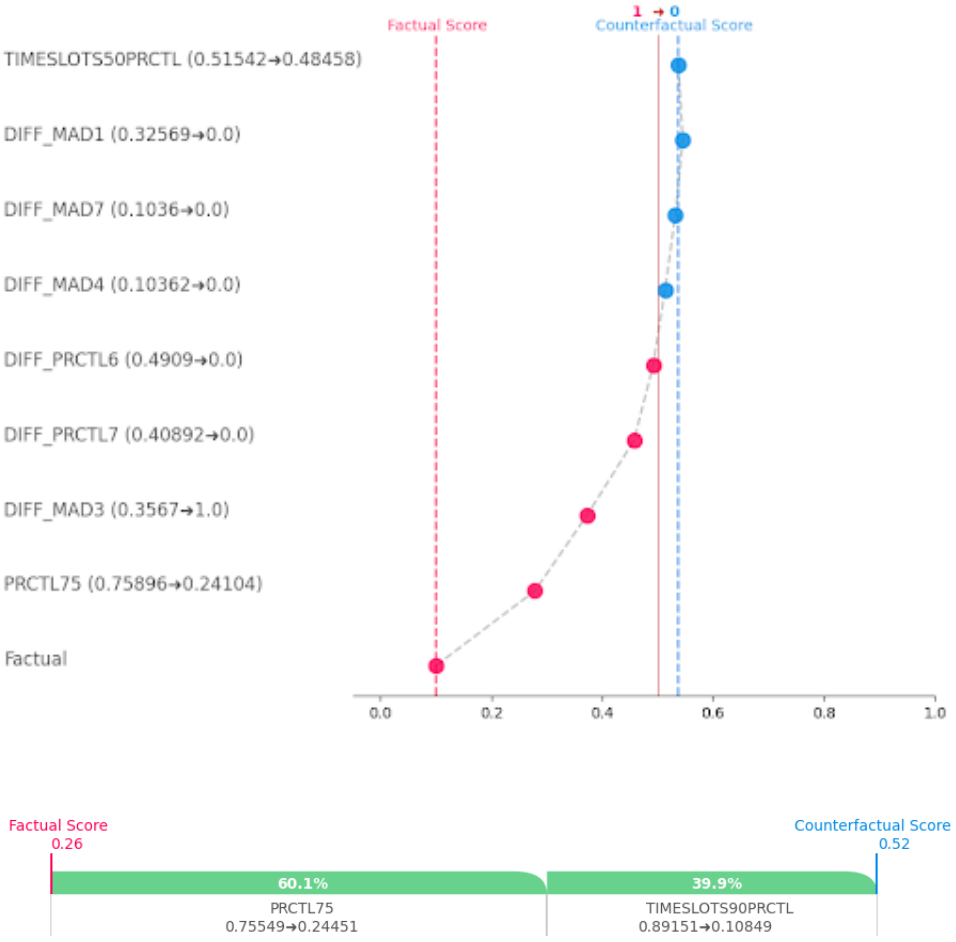


Figure 5: greedy chart e CounterShapley relativo ad un esempio con CustomIsolationForest

6.2.1 Analisi di grafici dello stesso modello

Osservando i grafici relativi a diverse istanze ma dello stesso modello è possibile osservare come, con entrambi i metodi di analisi, l'importanza relativa della features sia la medesima. Le stesse features si ripetono con un'alta frequenza a discapito di pochi esempi che ne considerano invece altre. Quanto detto è osservabile anche tra grafici differenti della stessa istanza di un singolo modello. Anche qui, come detto prima, è osservabile una forte

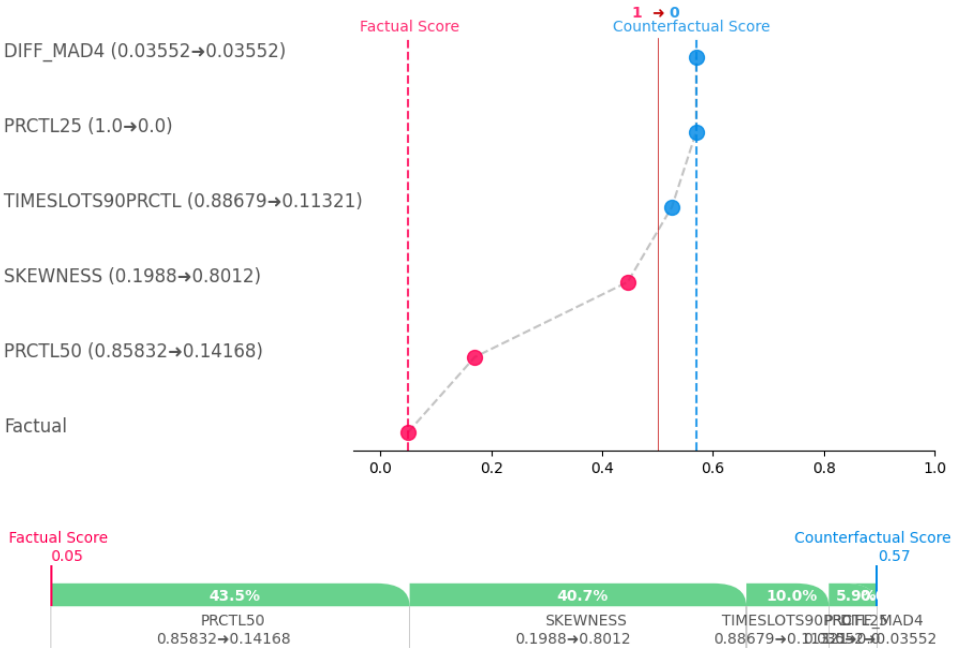


Figure 6: greedy chart e CounterShapley relativo ad un esempio con MLPClassifier

correlazione tra gli elementi presenti nel primo grafico e nel secondo. Infine è importante notare come l'ordine di contributo al cambiamento della classe rappresenti poi anche il cammino più greedy che permette il cambiamento di etichetta di un'istanza dati. La prima feature, in particolare, risulta essere sempre la stessa nei due grafici, come se il cambiamento di essa rappresentasse il contributo più importante al fine di determinare il cambiamento della classe all'istanza dei dati considerata.

6.2.2 Analisi di grafici di modelli differenti

Osservando i grafici di modelli differenti è invece possibile riscontrare alcune differenze, anche significative, nell'importanza relativa di alcuni parametri. Nell'analisi con *Isolation Forest*, la feature che risulta più importante è *PRCTL75*, mentre nell'analisi con *MLPClassifier* la feature che risulta più importante è *PRCTL50*. Nonostante queste discrepanze, altri parametri risultano avere una rilevanza confrontrabile per i due modelli.

6.3 Analisi Shapley values

Successivamente, è stata effettuata l'analisi tramite Shapley values con la libreria SHAP in maniera analoga all'analisi con i controfattuali, ossia sia per il modello MLP che per il modello Isolation Forest.

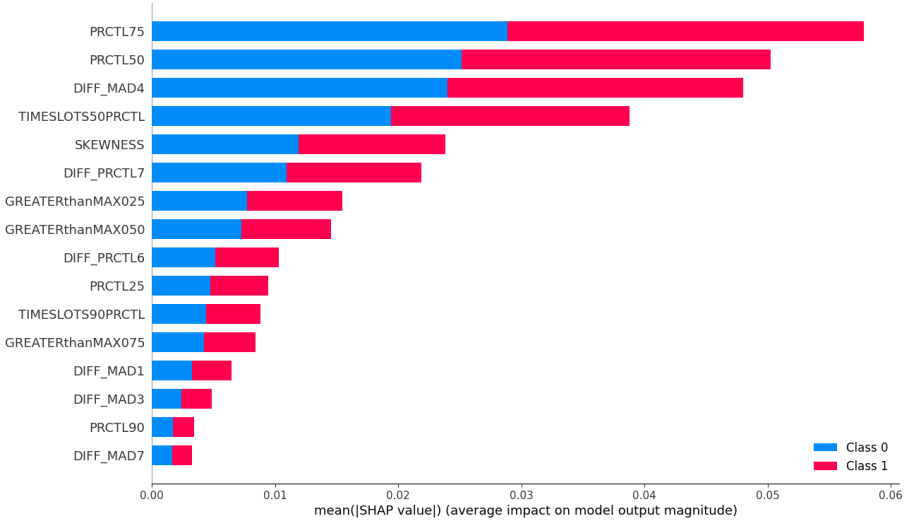


Figure 7: Grafico Shapley values di Isolation Forest

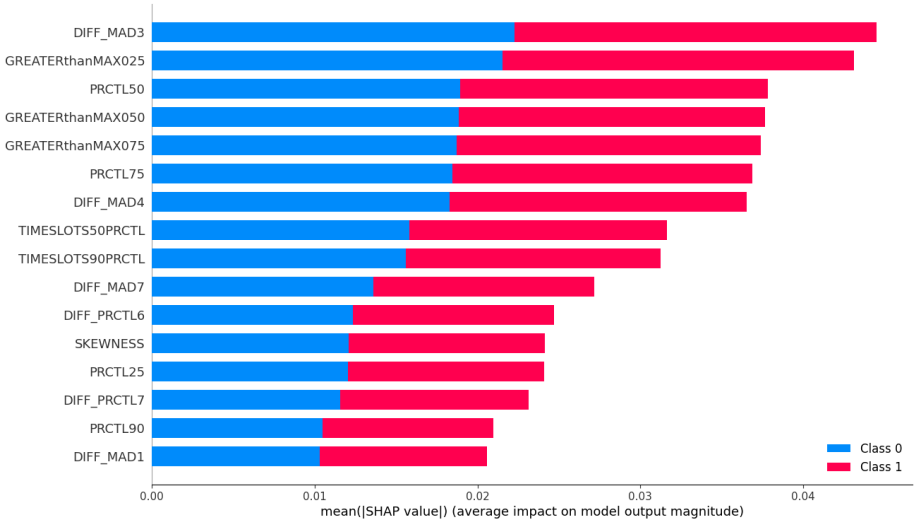


Figure 8: Grafico Shapley values di MLPClassifier

6.3.1 Analisi grafici Shapley Values

Attraverso questi due grafici è possibile trarre delle osservazioni analoghe a quanto fatto con l'analisi dei controfattuali: infatti anche qui è possibile osservare alcune features che vengono classificate con la stessa importanza, indipendentemente dal modello utilizzato (in questo caso PRCTL50), mentre altre features riscontrano un'importanza relativa molto differente a seconda del modello utilizzato per calcolare i valori. Anche con questa analisi, singolarmente ai due modelli le due features più importanti, come in precedenza, risultano essere PRCTL50 per MLPClassifier e PRCTL75 per Isolation Forest. Inoltre, a differenza del modello MLPClassifier, nell'analisi con Isolation Forest sembra che l'importanza relativa dei singoli parametri sia ristretto a poche variabili, ossia la determinazione principale dell'etichetta di una singola istanza dati sembra essere determinata principalmente da un numero di variabili molto minore rispetto a quella effettuata con MLPClassifier.

6.4 Analisi Confusion Matrix

Per studiare in maniera più approfondita la differenza di precisione tra i due classificatori, sono state calcolate le matrici di confusioni per entrambi i modelli, che permettono di analizzare in che modo viene classificata ogni singola etichetta.

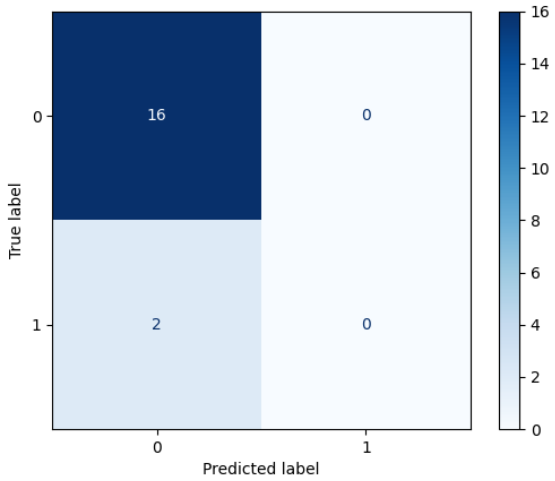


Figure 9: Confusion Matrix relativa a MLPClassifier

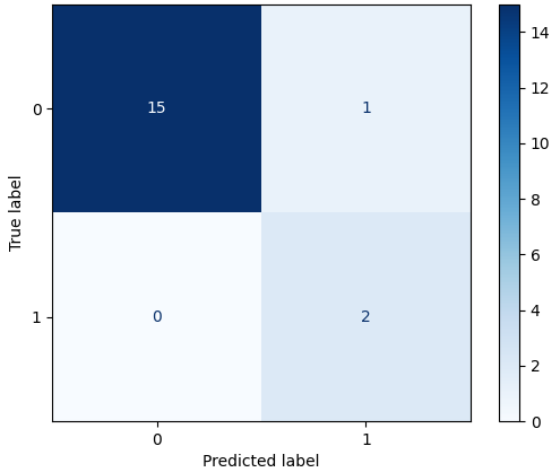


Figure 10: Confusion Matrix relativa a CustomIsolationForest

6.4.1 Osservazioni matrici di confusione

Dalle matrici di confusione è possibile osservare come, nel caso di Isolation Forest, le etichette considerate "anomale", cioè appartenenti alla classe 1 vengono classificate in maniera corretta. Al contrario, il modello MLPClassifier non riesce a classificarle in maniera corretta, riportanto tuttavia un alto grado di accuratezza dovuto al fatto dell'elevato sbilanciamento della classe 0 rispetto alla classe 1.

6.5 Analisi Wrapper CustomIsolationForest

Da questo momento in poi l'analisi si è concentrata maggiormente sullo studio del wrapper del modello Isolation Forest. L'obiettivo principale era capire e verificare se i metodi introdotti fossero compatibili anche con altre librerie di controfattuali, riuscendo a riscontrare valori simili con librerie differenti. Successivamente a questa analisi qualitativa, lo studio si è concentrato anche su un'analisi quantitativa, cercando di indagare, tramite alcune indici statistici, se i risultati ottenuti fossero in qualche modo correlati tra loro, al fine di dare una validità sistematica all'implementazione del wrapper del modello.

6.5.1 Analisi qualitativa

La libreria utilizzata in questa fase è stata la libreria *DiCE*, con cui sono stati testati la ricerca di controfattuali calcolando, tramite la funzione della libreria *global_feature_importance*, il contributo relativo di ogni features alla classe relativa a quei valori utilizzando esclusi-

vamente il modello CustomIsolationForest.

- SKEWNESS: 0.727;
- PRCTL75: 0.655;
- DIFF_MAD1: 0.427;
- PRCTL25: 0.400;
- TIMESLOTS50PRCTL: 0.391;
- GREATERthanMAX025: 0.373;
- PRCTL90: 0.373;
- TIMESLOTS90PRCTL: 0.364;
- GREATERthanMAX075: 0.364;
- DIFF_MAD7: 0.327;
- DIFF_MAD3: 0.291;
- PRCTL50: 0.282;
- GREATERthanMAX050: 0.273;
- DIFF_PRCTL6: 0.255;
- DIFF_MAD4: 0.227;
- DIFF_PRCTL7: 0.227;

Questi indici identificano i contributi relativi delle singole features alle classi del dataset oggetto di studio, potendo evincere quindi che il metodo scritto nel modello wrapper (*predict_proba*) sia compatibile anche con questa altra libreria per il calcolo dei controfattuali. I contributi individuali delle features sono confrontabili con i valori calcolati nel precedente report, come è possibile osservare, ad esempio, tramite il parametro PRCTL75 ad esempio. Ciò, oltre a fornirci un'indicazione positiva sull'implementazione del modello, verifica ancora una volta tramite un metodo differente l'importanza maggiore di alcune feature rispetto ad altre. In particolare, si può osservare, con librerie differenti, una forte

correlazione tra i singoli contributi se osservati tramite un unico modello. Al contrario si nota una correlazione più bassa ma comunque presente, se analizziamo i singoli contributi con due modelli differenti (nel nostro caso MLPClassifier e Isolation Forest).

6.5.2 Analisi quantitativa

Infine, l'analisi si è concentrata sull'utilizzo sistemico del wrapper descritto precedentemente, per confrontrare dei risultati sperimentali rispetto anche ad altre misure di feature importance, come "*permutation importance*". Come nel precedente report, tramite la libreria DiCE sono stati testati la ricerca di controfattuali utilizzando la funzione della libreria *global_feature_importance* per calcolare il contributo relativo di ogni features alla classe relativa a quei valori. A questo punto tramite la funzione *permutation_importance*, è stato anche qui misurato per ogni feature un parametro di importanza, dato dalla funzione score di CustomIsolationForest. Da questi due vettori risultanti (un vettore rappresenta l'importanza relativa di ogni feature ottenuto tramite la funzione global_feature_importance e l'altro tramite la funzione permutation_importance) sono stati calcolati lo *Spearman correlation* e il *p-value*, indici statistici che permettono di osservare quanto simili siano le due misure. Questa operazione (calcolo dei due vettori e calcolo dei due indici) è stata iterata 10 volte sullo stesso dataset (*x_test*, *y_test*). Infine, sono stati calcolate le medie e le deviazioni standard delle 10 misurazioni dello Spearman correlation e del p-value.

media Spearman correlation: **0.498**
dev standard Spearman Correlation: **0.072**

media p-value: **0.061**
dev standard p-value: **0.045**

Il valore medio trovato dell'indice Spearman correlation ci fornisce un'informazione secondo la quale è presente una relazione di monotonia (seppur non troppo forte) tra i due vettori trovati. Ciò è indice del fatto che anche con metriche diverse il contributo relativo delle singole feature è confrontabile. La soglia di test del p-value utilizzata è stata 0.05, osservando quindi che la media del p-value calcolata è leggermentente maggiore della soglia test. Tuttavia, possiamo considerare tale valore parzialmente accettabile visti gli obiettivi della tesi e il grado di accuratezza dell'analisi svolta durante il lavoro. Pertanto,

sotto queste ipotesi, non è possibile escludere il fatto che i due vettori calcolati siano correlati tra loro, indicando un'altra volta che potrebbe esistere una correlazione tra i due vettori e andando quindi ad offrire un'ennesima valutazione positiva dell'implementazione del wrapper.

Capitolo 7

Conclusioni

Grazie ai due diversi modelli implementati (MLPClassifier e CustomIsolationForest) abbiamo potuto costruire due classificatori con buone prestazioni in termini di accuratezza e rilevamento delle anomalie.
Accuracy score MLPClassifier: 0.88889
Accuracy score Isolation Forest: 0.94444
C'è tuttavia da dire che l'alto sbilanciamento del numero di esempi della classe 0 (40 esempi di dati non anomali) rispetto al numero di esempi della classe 1 (4 esempi di dati anomali) potrebbe aver causato alcuni errori nella fase di addestramento, portando i modelli a classificare con probabilità molto più alta gli input nella classe 0. Questa fenomeno è stato possibile osservarlo anche attraverso le matrici di confusione, confermando che il modello MLPClassifier non riesca a classificare correttamente le etichette anomale. La precisione che ne risulta, a livello di accuratezza, è comunque elevata a causa dello sbilanciamento degli esempi per singola classe. Sicuramente ciò potrebbe rivelarsi un problema dato che nell'analisi del rilevamento delle anomalie l'interesse maggiore è scoprire quando un insieme di dati è classificabile nella classe 1, rispetto a individuare un gruppo di dati appartenenti alla classe 0. Le tecniche di XAI utilizzate ci hanno fornito dei buoni risultati se contestualizzate a valori diversi e metriche diverse ma applicate allo stesso modello. Al contrario, i risultati di tali tecniche sono state peggiori confrontando i risultati dello stesso dataset ma con due modelli differenti. Anche questa problematica potrebbe dipendere fortemente dalla natura dei dati e dalla scarsità di esempi per istanze di tipo 1. Ne risultano comunque alcune feature più importanti di altre, anche se analizzate con modelli differenti. Da segnalare le variabili PRCTL50 e PRCTL75 che risultano avere un ordine di importanza elevata con entrambi i modelli. L'analisi qualitativa del wrapper del modello Isolation Forest ha riportato valori confrontabili con quanto osservato precedentemente, continuando ad osservare dei buoni risultati nell'importanza

relativa delle variabili del dataset seppur analizzate con librerie differenti. Questo sicuramente fornisce una prima validazione positiva del wrapper implementato e del suo utilizzo in questi contesti di XAI. Il grado di accuratezza dell'implementazione del modello è stato poi anche verificato con un'analisi più sistematica di CustomIsolationForest. Infatti, sotto l'ipotesi di lavoro di di condurre un'analisi quantitativa preliminare del wrapper del modello, è stato possibile dimostrare, tramite due indici statistici come Spearman correlation (media di 0.498) e p-value (media di 0.061) che effettivamente gli output di tali metodo risultassero "correlati", seppur in maniera non troppo forte. Possiamo quindi concludere che le ipotesi di lavoro iniziali sono state parzialmente soddisfatti, riuscendo ad applicare le tecniche di XAI ad un contesto di analisi dati di anomaly detection, riscontrando tuttavia alcune criticità dovute principalmente alla natura stessa del dataset, al suo numero di esempi e alla sua distribuzione delle classi. L'analisi è stata portata a termine anche grazie all'estensione di uno dei modelli analizzati, senza il quale non avremmo potuto portare a termine il lavoro e non avremmo potuto confrontare due classificatori differenti. La validità parziale dell'implementazione è stata poi anche validata tramite due analisi, una qualitativa e una quantitativa. La prima analisi ci ha mostrato come la soluzione fornita potesse funzionare anche con librerie differenti. La seconda analisi, invece, ha permesso di riscontrare una correlazione statistica tra i valori calcolati, tramite alcuni indici come Spearman Correlation e p-value, i cui risultati hanno soddisfatto le nostre ipotesi di attese iniziali.

7.1 Lavori futuri

L'analisi futura che si consiglia di fare è quella di testare i due modelli tramite un dataset con un numero di esempi maggiore, e soprattutto con una distribuzione tra classi molto meno sbilanciata, anche se è importante ricordare come nel contesto di anomaly detection la presenza di anomalie è un evento per natura raro e quindi andrà sempre considerato una sbilanciamento di classi in un determinato dataset. Altri studi potrebbero invece considerare metriche diverse da quelle utilizzate in questo studio, in modo da testare il wrapper del modello anche in situazioni differenti da quelle analizzate in questo lavoro e in maniera più sistematica, in modo da poter ottenere gradi di accuratezza maggiore negli indici statistici calcolati.

Capitolo 8

Appendice A

In questa sezione viene riportato la documentazione del software prodotto durante tutto il lavoro di tesi. Viene anche riportato il codice relativo al wrapper del modello Isolation Forest.
Il progetto, creato in *PyCharm*, si suddivide in alcune cartelle:

- **classes**: contiene l'implementazione del wrapper del modello di Isolation Forest e la classe Dataset, implementata per gestire in maniera più efficiente la gestione del dataset;
- **config**: contiene il file config.json, che permette di selezionare il nome del dataset su cui andare a lavorare e il nome delle target features del dataset e il file modelConfiguration.json, che contiene gli iperparametri del modello MLPClassifier;
- **initialization**: contiene il file initializationCustomIsolationForest.py, in cui è presente l'inizializzazione del mio modello attraverso i parametri utilizzati durante tutta la fase di analisi dati descritta nei report precedenti e initializationDataset.py, che contiene l'inizializzazione della classe Dataset;
- **output**: contiene i file prodotti dagli script;
- **resources**: contiene il dataset analizzato;
- **script**: contiene gli script utilizzati durante il progetto, MLP_training_accuracy.py e CustomIsolationForest_accuracy.py contengono l'addestramento ed il calcolo dello score dei due modelli. MLPClassifier.py permette di esportare il modello trovato con i propri iperparametri per utilizzarlo in altri script ed, infine, testDataset.py contiene l'esecuzione dei metodi della classe Dataset per effettuare l'analisi descritta nei capitoli precedenti.

Listing 1: Codice CustomIsolationForest

```
import csv
import dice_ml
from sklearn.ensemble import IsolationForest
import numpy as np
from sklearn.metrics import accuracy_score

class CustomIsolationForest(IsolationForest):
    def __init__(self, n_estimators=100, max_samples='auto', contamination='
        ↪ auto',
                 max_features=1.0, bootstrap=False, n_jobs=None, random_state=
                     ↪ None,
                 verbose=0, warm_start=False,
                 distance_min=None, distance_max=None):
        super().__init__(n_estimators=n_estimators, max_samples=max_samples,
                         contamination=contamination, max_features=max_features,
                         bootstrap=bootstrap, n_jobs=n_jobs, random_state=
                             ↪ random_state,
                         verbose=verbose, warm_start=warm_start)
        self.distance_min = distance_min
        self.distance_max = distance_max

    def init_predict_proba(self, values):
        anomaly_score = self.decision_function(values)
        self.distance_max = anomaly_score.max()
        self.distance_min = anomaly_score.min()

    def predict_proba(self, value):
        class_0_probability = (self.decision_function(value) - self.
            ↪ distance_min) / (self.distance_max - self.distance_min)
        class_1_probability = 1 - class_0_probability
        prob_classes = np.vstack((class_0_probability, class_1_probability)).T
        return prob_classes

    def score(self, x_test, y_test):
        y_pred = 1 - (self.predict(x_test) + 1) // 2
        return accuracy_score(y_test, y_pred)
```

Listing 2: Codice classe Dataset

```
from cfnow import find_tabular
from sklearn.model_selection import train_test_split
import pandas as pd
import shap
import warnings

warnings.filterwarnings("ignore", message="X does not have valid feature names , but MLPClassifier was fitted with feature names")
warnings.filterwarnings("ignore", message="X does not have valid feature names , but CustomIsolationForest was fitted with feature names")

class Dataset:
    def __init__(self, dataset_name, target_feature_name):
        self.dataset = pd.read_csv("../resource/" + dataset_name)
        self.target = self.dataset[target_feature_name]
        self.x_train, self.x_test, self.y_train, self.y_test = train_test_split(self.dataset, self.target, test_size=0.4, random_state=42)
        self.x_train = self.x_train.drop(target_feature_name, axis=1)
        self.x_test = self.x_test.drop(target_feature_name, axis=1)

    def get_dataset_train(self):
        return pd.concat([self.x_train, self.y_train], axis=1)

    def get_dataset_test(self):
        return pd.concat([self.x_test, self.y_test], axis=1)

    def get_X(self):
        return self.dataset.drop('ANOMALOUS', axis=1)

    def get_shapley_values(self, model):
        explainer = shap.KernelExplainer(model.predict_proba, self.x_train)
        shap_values = explainer.shap_values(self.x_test)
        shap.summary_plot(shap_values, self.x_test)

    def get_greedy_chart(self, model, row):
        cols_numericals = [column for column in self.dataset.select_dtypes(include=['int64']).columns]

        cf_res = find_tabular(
            factual=self.x_test.iloc[row],
            feat_types={c: 'num' if c in cols_numericals else 'cat' for c in self.get_X().columns},
            has_ohe=True,
            model_predict_proba=model.predict_proba,
            limit_seconds=60)

        cf_res.generate_counterplots(0).greedy('../output/greedy.png')

    def get_counterplots(self, model, row):
        cols_numericals = [column for column in self.dataset.select_dtypes(include=['int64']).columns]
        cf_res = find_tabular(
            factual=self.x_test.iloc[row],
            feat_types={c: 'num' if c in cols_numericals else 'cat' for c in self.get_X().columns},
```

```
            has_ohe=True,
            model_predict_proba=model.predict_proba,
            limit_seconds=60)
        cf_res.generate_counterplots(0).countershapley('../output/
            ↪ countershapley.png')
```

Listing 3: inizializzazione CustomIsolationForest

```
from classes.CustomIsolationForest_class import CustomIsolationForest
from initialization.initialization_Dataset import Dataset

CIF_model = CustomIsolationForest()
CIF_model = CIF_model.fit(Dataset.x_train, Dataset.y_train)
CIF_model.init_predict_proba(Dataset.x_train)
```

Listing 4: inizializzazione classe Dataset

```
from classes.Dataset_class import Dataset
import json

with open("../config/config.json", "r") as config_file:
    config = json.load(config_file)
dataset_name = config["dataset_name"]
target_feature_name = config["target_feature"]
Dataset = Dataset(dataset_name, target_feature_name);
```

Listing 5: Accuracy CustomIsolationForest

```
from initialization.initialization_CustomIsolationForest import CIF_model
from initialization.initialization_Dataset import Dataset
print(CIF_model)
print(CIF_model.predict_proba(Dataset.x_test))
print(CIF_model.predict(Dataset.x_test))
print(Dataset.y_test)
print(CIF_model.score(Dataset.x_test, Dataset.y_test))
```

Listing 6: Accuracy MLPClassifier

```
import json
from sklearn.metrics import accuracy_score
from sklearn.neural_network import MLPClassifier
from initialization.initialization_Dataset import Dataset
import matplotlib.pyplot as plt

with open('../config/modelConfiguration.json', 'r') as json_file:
    model_params = json.load(json_file)
clf = MLPClassifier(hidden_layer_sizes=model_params['hidden_layer_sizes'],
                    activation=model_params['activation'],
                    solver=model_params['solver'],
                    alpha=model_params['alpha'],
                    learning_rate=model_params['learning_rate']
                    )
clf.fit(Dataset.x_train, Dataset.y_train)
labels = clf.predict(Dataset.x_test)
accuracy = accuracy_score(Dataset.y_test, labels)
```

```
print(Dataset.y_test, labels)
print("Accuracy:", accuracy)
```

Listing 7: test classe Dataset

```
from initialization.initialization_CustomIsolationForest import CIF_model
from initialization.initialization_Dataset import Dataset
from script.MLPClassifier import clf

Dataset.get_shapley_values(clf)
Dataset.get_shapley_values(CIF_model)
Dataset.get_greedy_chart(clf, 0)
Dataset.get_greedy_chart(CIF_model, 0)
Dataset.get_counterplots(clf, 0)
Dataset.get_counterplots(CIF_model, 0)
CIF_model.test_spearman(Dataset);
```

Capitolo 9

Appendice B

In questa parte sono riportati i grafici greedy chart e counterShapley calcolati per ogni input e per ognuno dei due modelli.

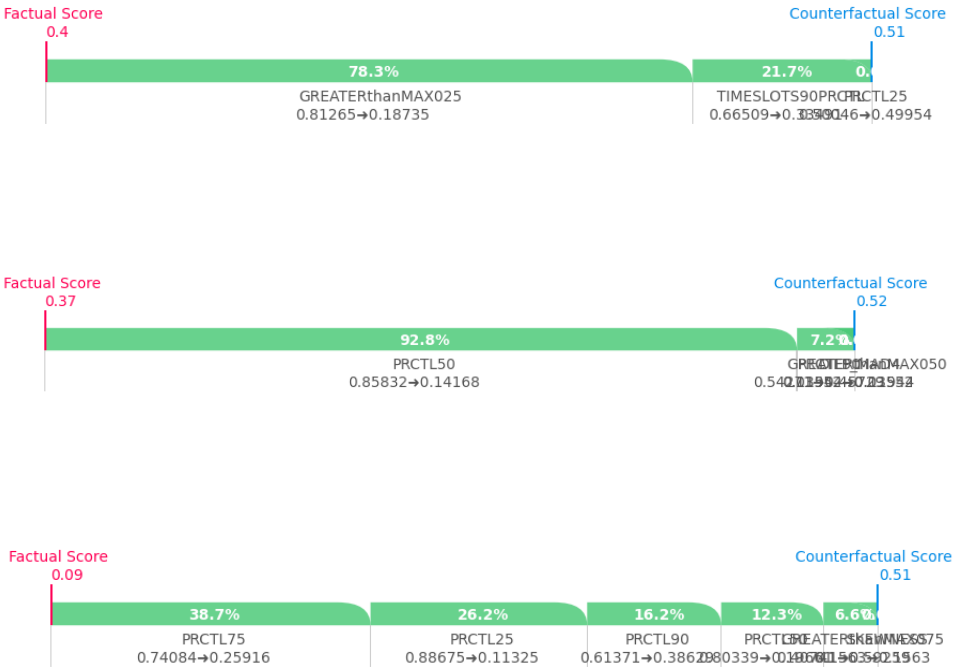


Figure 11: grafici CounterShapley per il modello IsolationForest

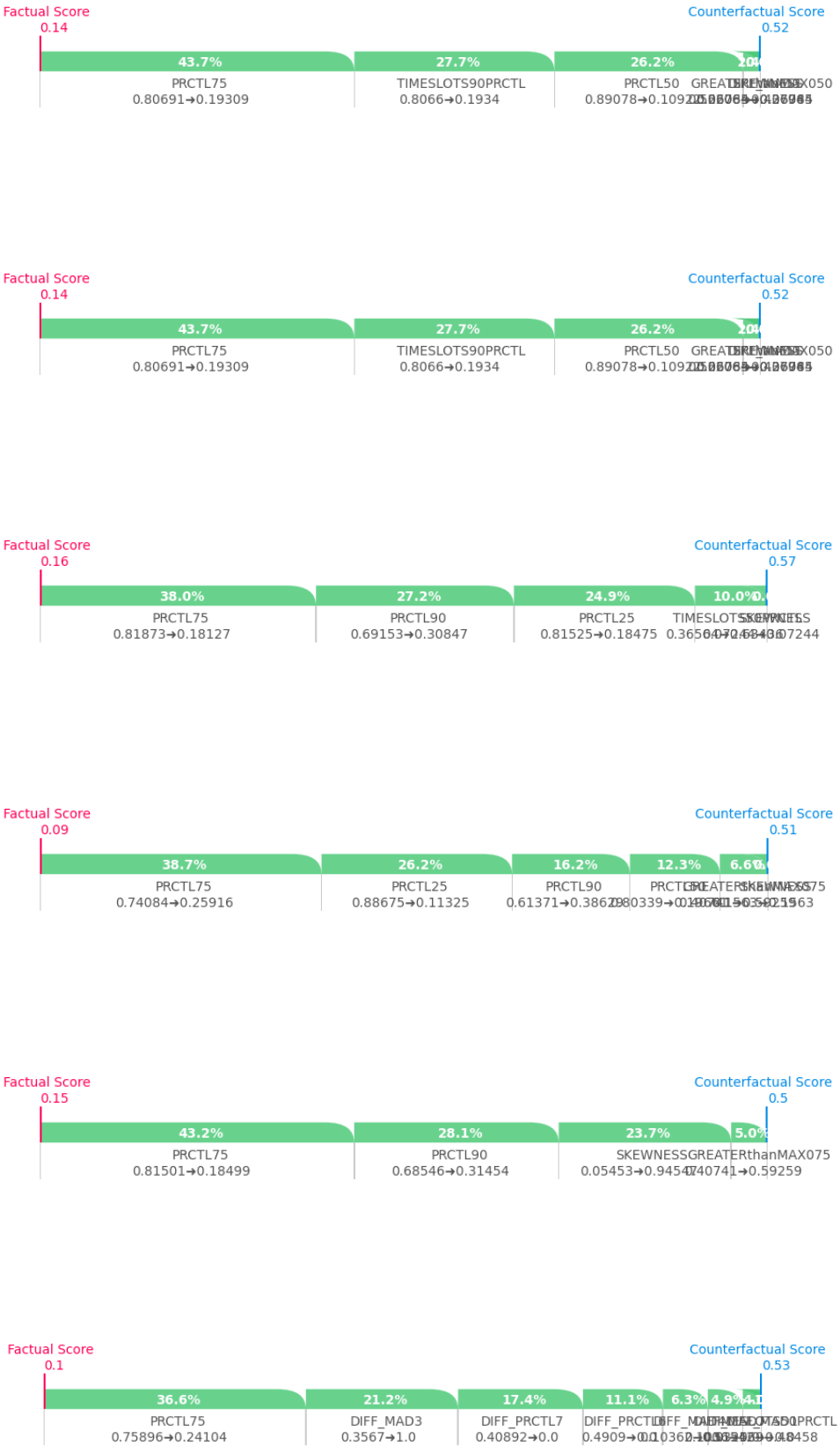


Figure 12: grafici CounterShapley per il modello IsolationForest

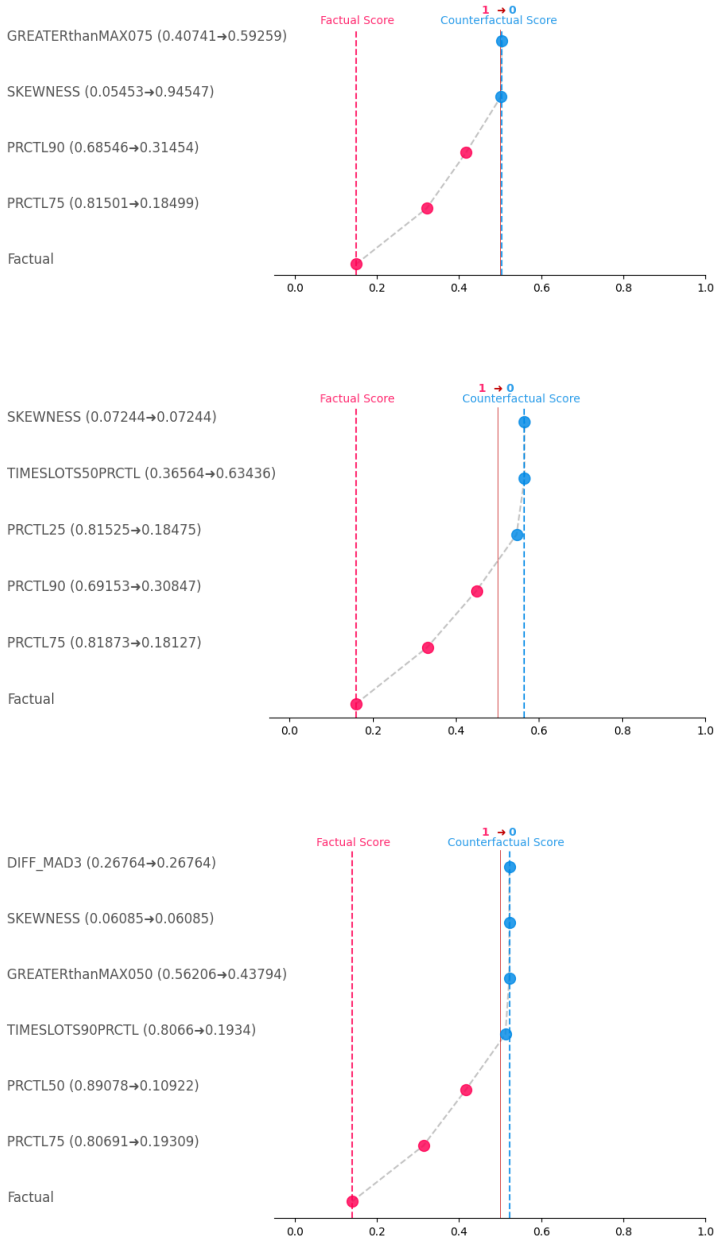


Figure 13: grafici greedy chart per il modello IsolationForest

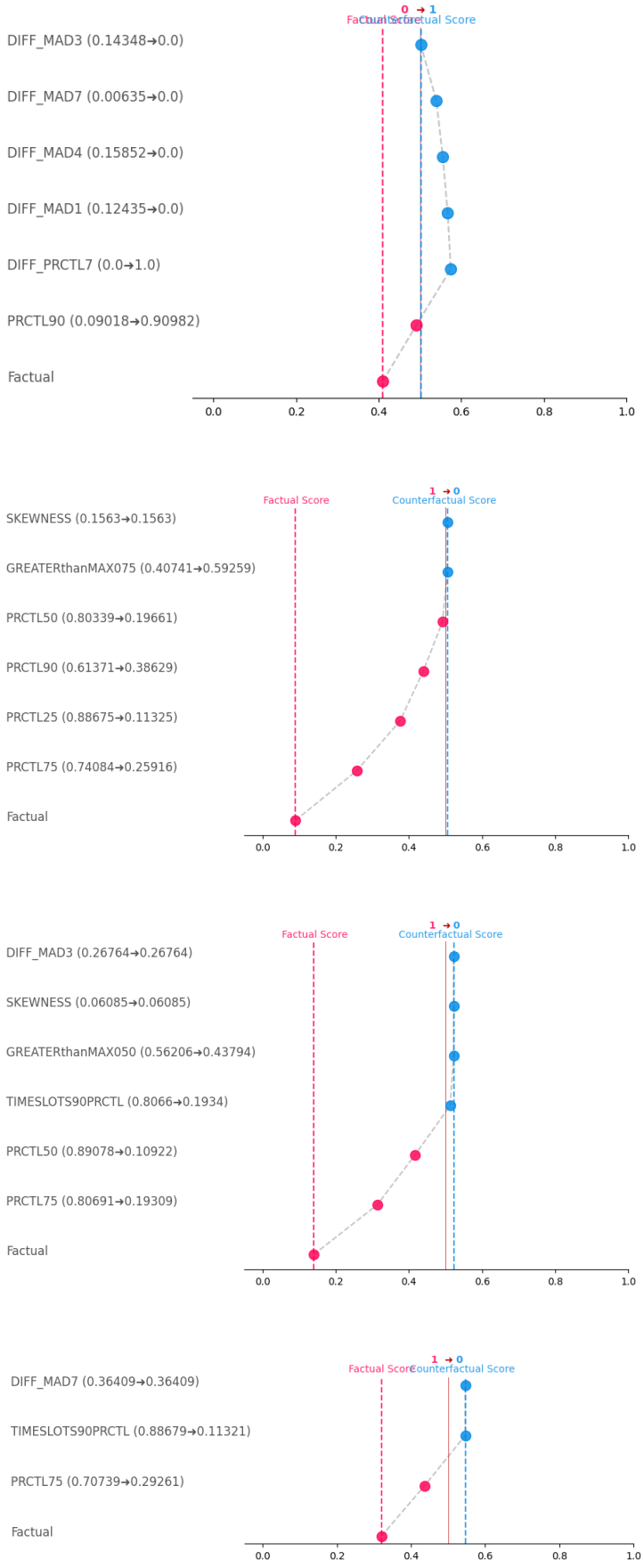


Figure 14: grafici greedy chart per il modello IsolationForest

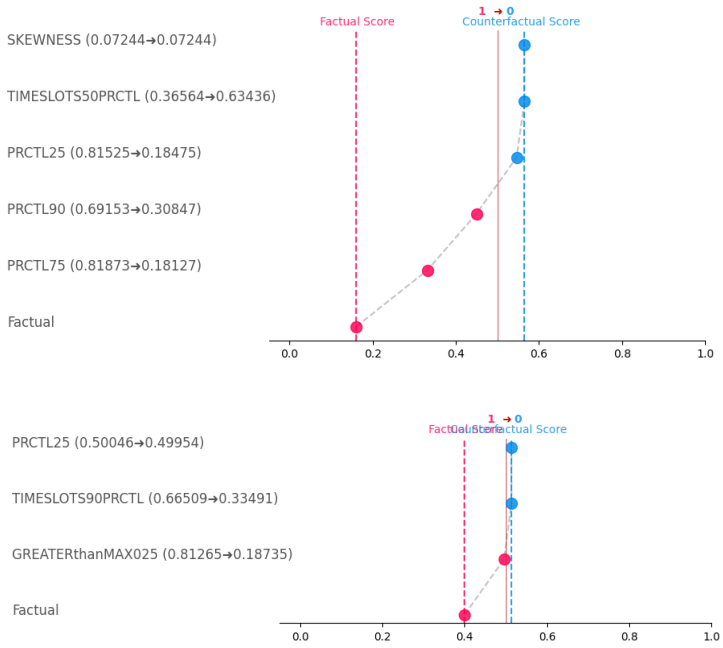


Figure 15: grafici greedy chart per il modello IsolationForest

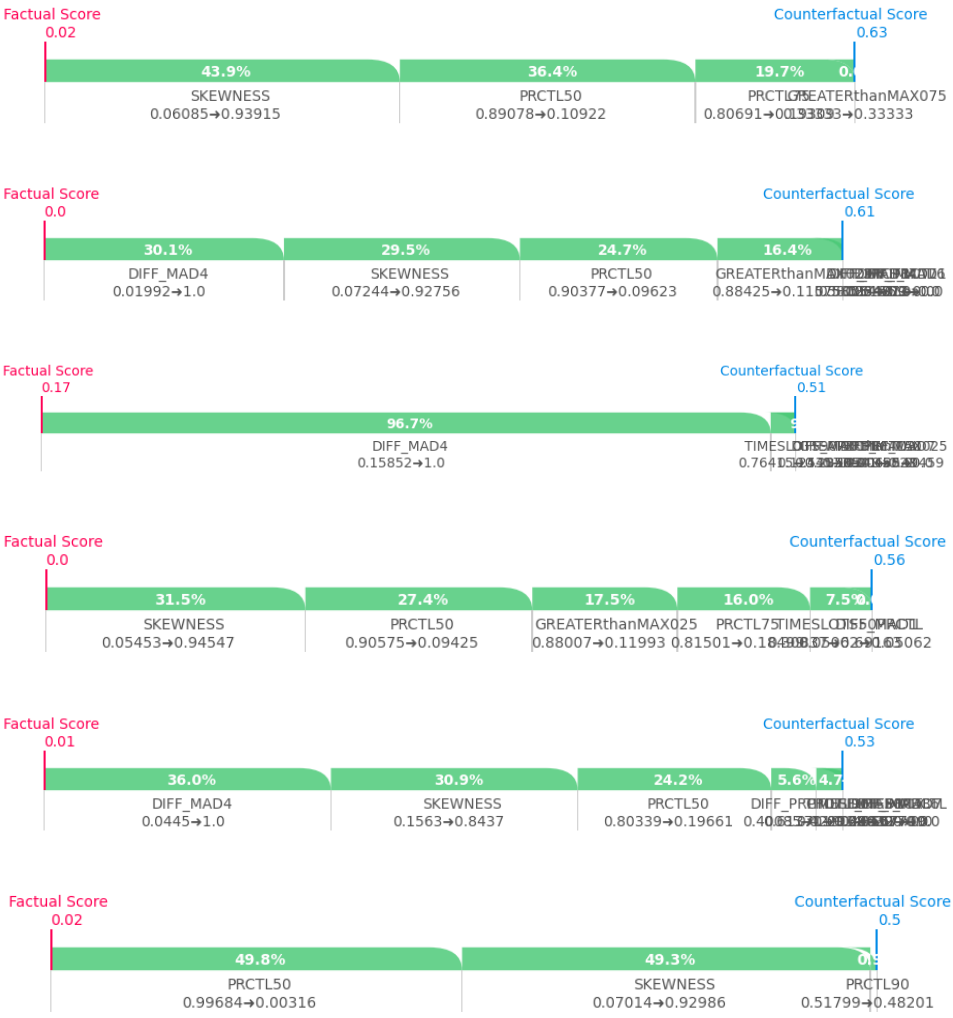


Figure 16: grafici CounterShapley per il modello MLPClassifier

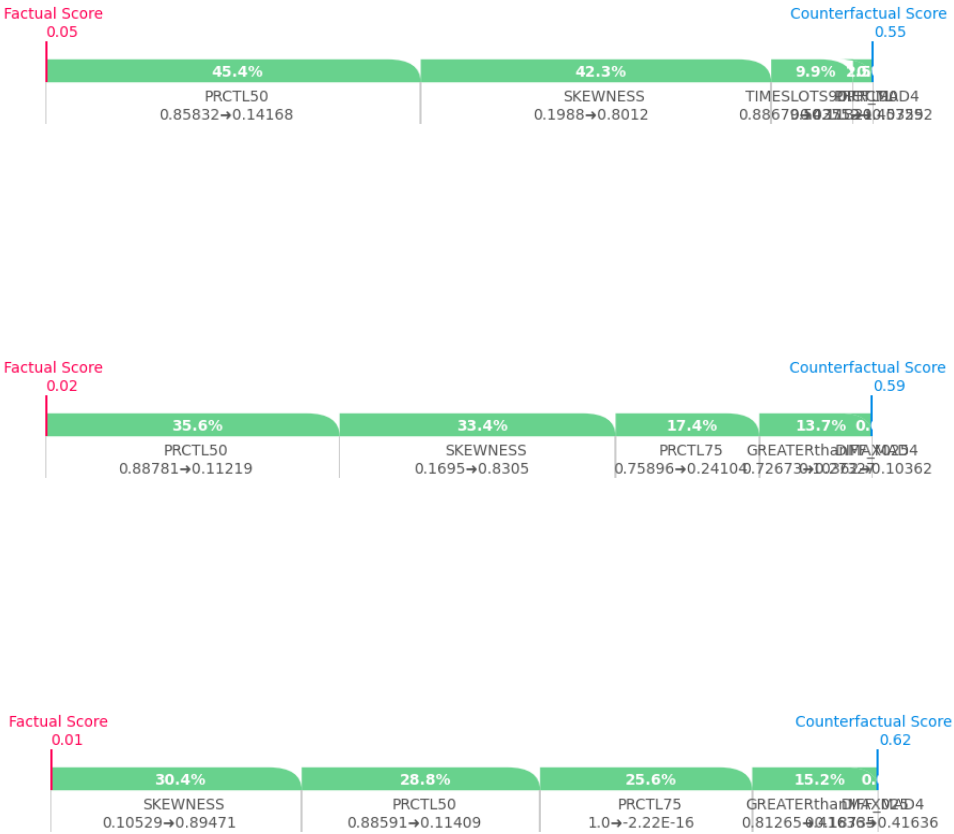


Figure 17: grafici CounterShapley per il modello MLPClassifier

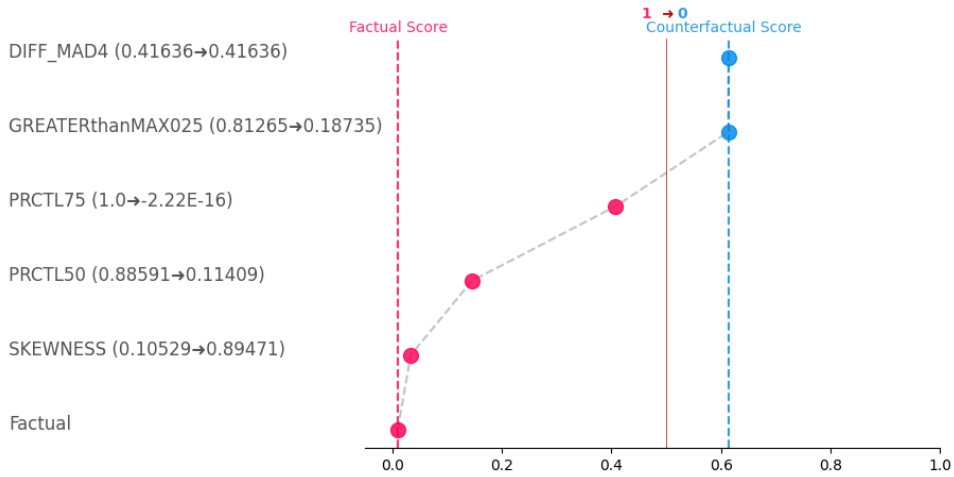
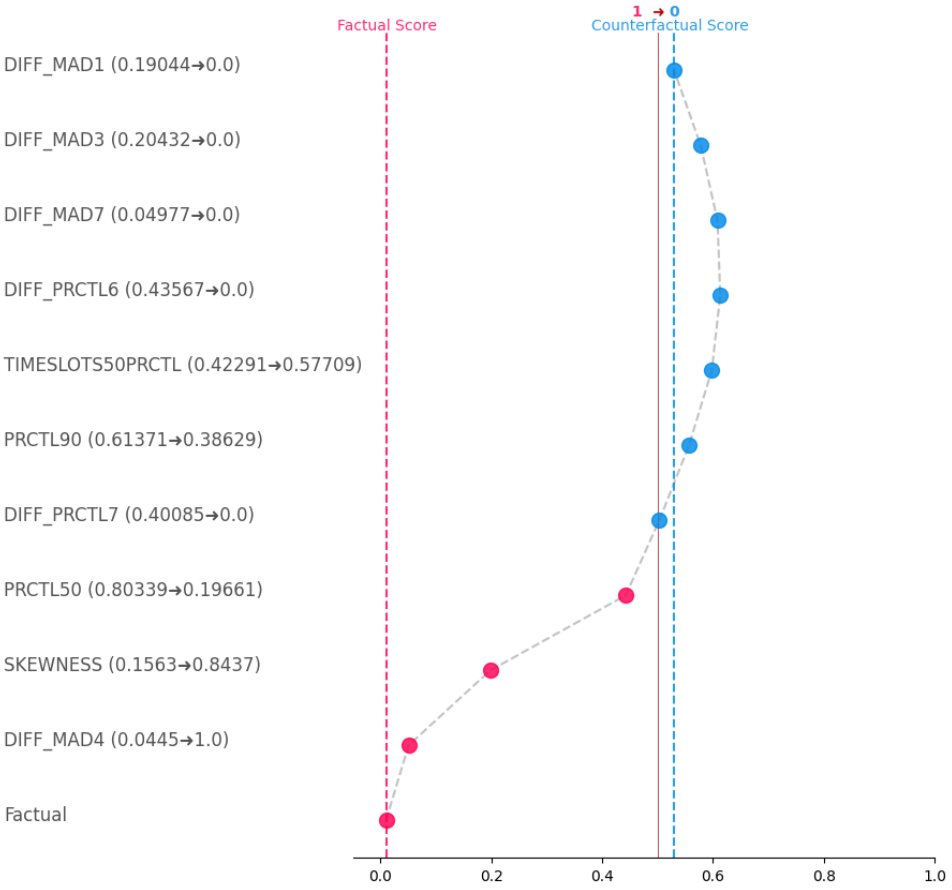


Figure 18: grafici greedy chart per il modello MLPClassifier



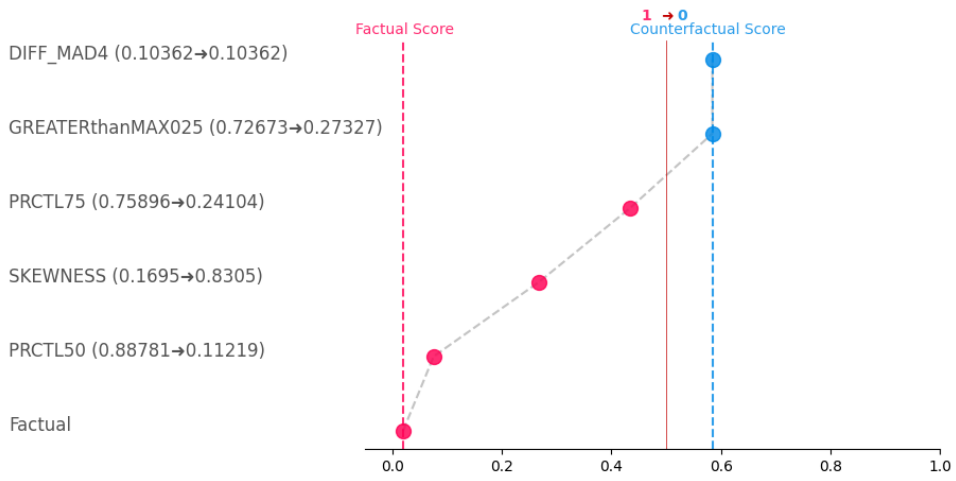
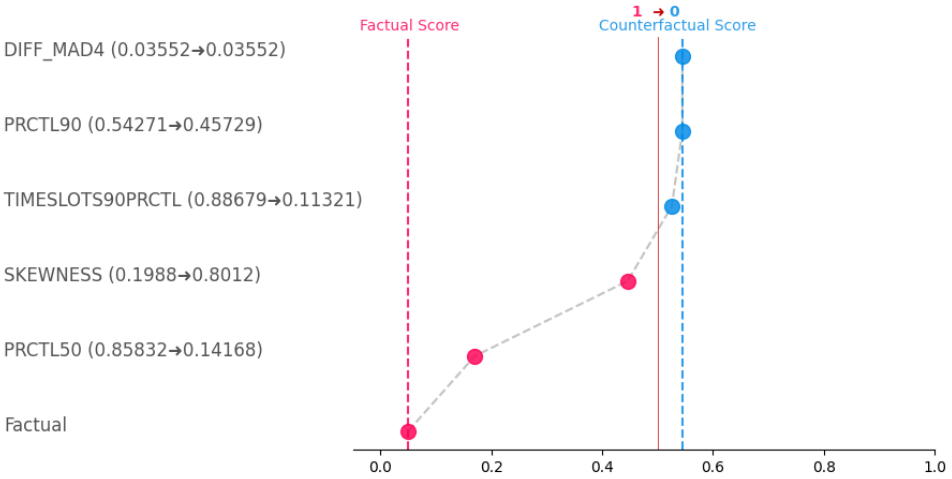
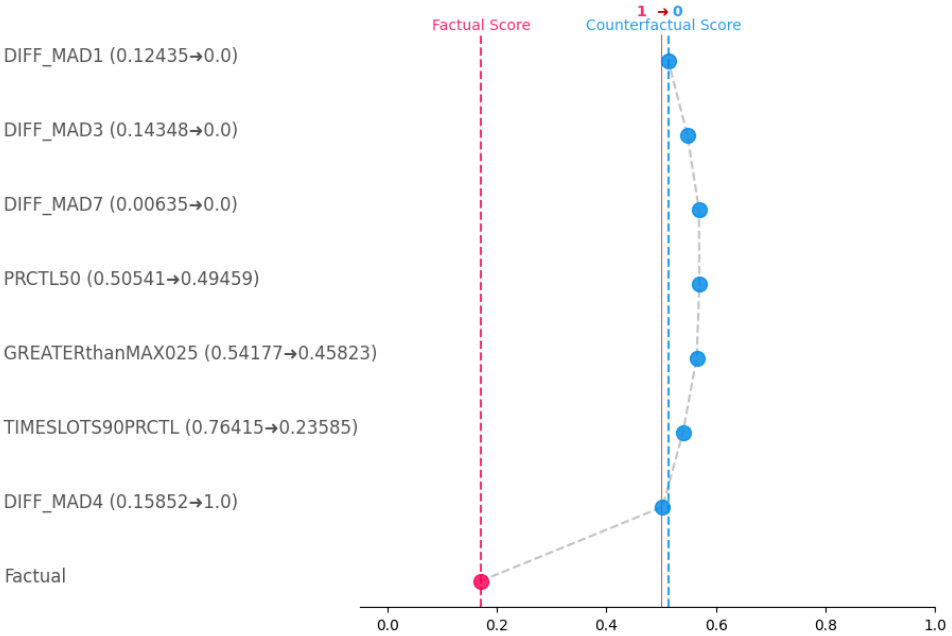


Figure 19: grafici greedy chart per il modello MLPClassifier





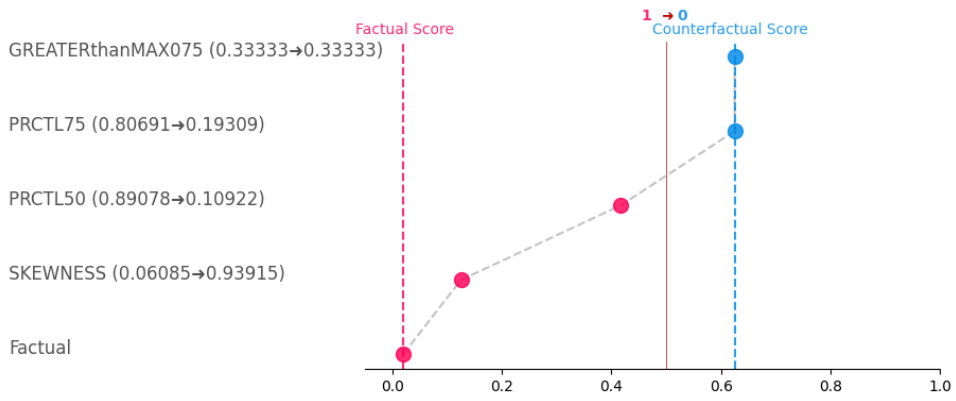
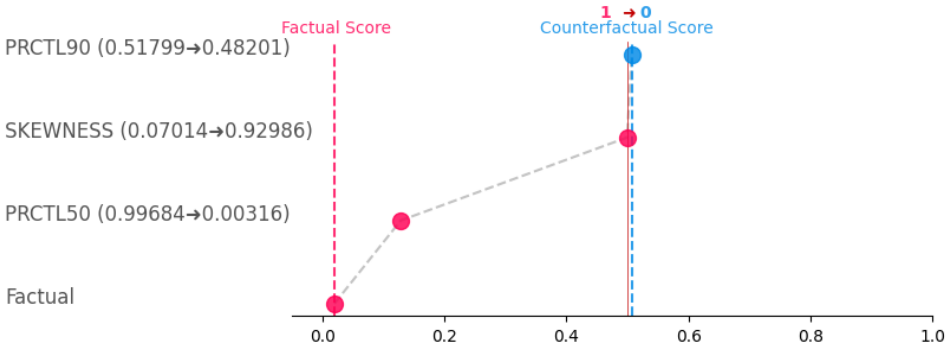
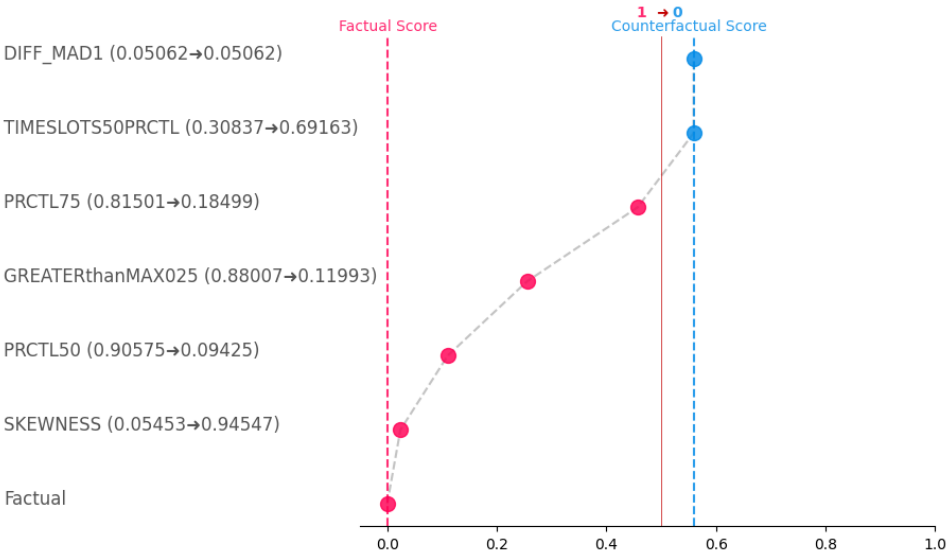


Figure 20: grafici greedy chart per il modello MLPClassifier





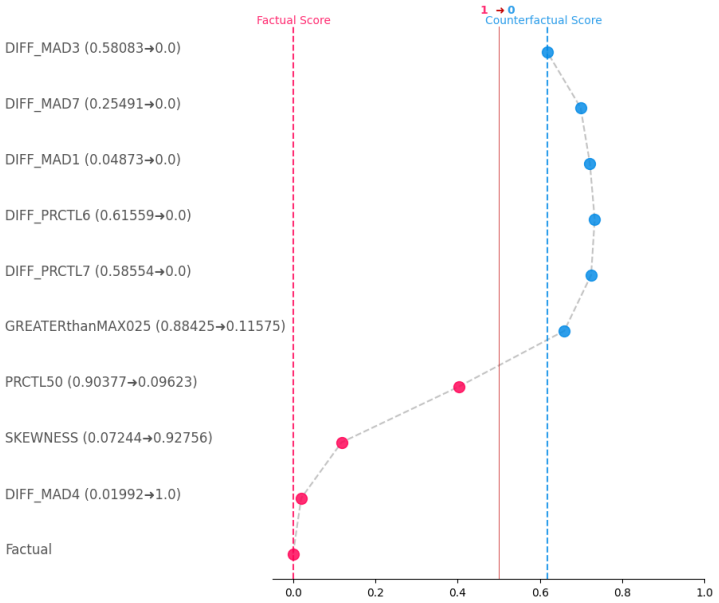


Figure 21: grafici greedy chart per il modello MLPClassifier

Capitolo 10

Bibliografia

[1] What is AI? — IBM - ibm.com https://www.ibm.com/topics/artificial-intelligence

[2] What is XAI? — IBM - ibm.com https://www.ibm.com/topics/explainable-ai

[3] I. Ahmed, G. Jeon and F. Piccialli, "From Artificial Intelligence to Explainable Artificial Intelligence in Industry 4.0: A Survey on What, How, and Where," in IEEE Transactions on Industrial Informatics, vol. 18, no. 8, pp. 5031-5042, Aug. 2022, doi: 10.1109/TII.2022.3146552.

[4] Mirka Saarela, Susanne Jauhiainen. Comparison of feature importance measures as explanations for classification models. February 2021.

[5] Sharma, Jeetesh Mittal, Murari Soni, Gunjan. (2023). Explainable artificial intelligence (XAI) enabled anomaly detection and fault classification of an industrial asset. 10.21203/rs.3.rs-2780708/v1.