POLITECNICO DI TORINO

Corso di Laurea
in Matematica per l'Ingegneria

Tesi di Laurea

I Valori di Shapley per una Interpretabilità Model-Agnostic

Relatore

Prof. Francesco Vaccarino

Candidato

Marco Mungai Coppolino

Anno Accademico 2023-2024

Sommario

La tesi “I Valori di Shapley per una Interpretabilità Model-Agnostic” affronta il tema di come la crescente complessità dei modelli di machine learning abbia generato negli ultimi anni la necessità di strumenti in grado di spiegarne le previsioni. Nel contesto di questo bisogno, l'elaborato presenta un’analisi approfondita di SHAP (SHapley Additive exPlanations) 30[1], una raccolta di algoritmi interpretativi basati sui Valori di Shapley, un risultato della Teoria dei Giochi. In particolare, la tesi si concentra sullo studio delle spiegazioni Model-Agnostic, ovvero su quei metodi capaci di interpretare i risultati di qualsiasi modello, anziché specializzarsi su uno specifico. Il testo si apre con un'introduzione che delinea l'importanza di questa decisione, e poi, attraverso un esempio pratico di gioco elementare, viene esposta la teoria alla base dei Valori di Shapley. Successivamente l’attenzione si sposta momentaneamente sull’analisi di un database contenente dei campioni di “vinho verde” portoghese, a cui è associato un problema di classificazione. L’obiettivo è quello di riuscire a prevedere la qualità dei campioni di vino basandosi sui risultati di analisi chimico-fisiche contenuti nel dataset. Come classificatore è stato scelto di allenare una Support Vector Machine (SVM), e nella tesi vengono riportati i risultati di una ottimizzazione di iperparametri, svolta con l’intento di migliorare l’accuratezza dell’algoritmo. Dopo tale digressione si ritorna a parlare di SHAP prima mostrando come i risultati della teoria possano essere adattati ai modelli di Machine Learning e poi spiegando il funzionamento dei metodi che si è scelto di adottare. Nell’ultimo capitolo, infine, viene riportata l’interpretazione della SVM, includendo anche un approccio per gestire la correlazione tra le feature del dataset, uno dei problemi più comuni di SHAP.

Indice

Introduzione

L'evoluzione dell'intelligenza artificiale (AI) è stata una delle trasformazioni più significative e pervasive degli ultimi anni, influenzando in maniera sempre più profonda la nostra vita quotidiana L'aumento di popolarità di modelli generativi avanzati, come ChatGPT, ha innescato una corsa da parte delle aziende per finanziare tali tecnologie, con l'intento di ottenere un vantaggio sulla competizione. Questo fenomeno riflette il ruolo centrale che l'IA sta assumendo nel ridefinire e ottimizzare i processi di produzione e i servizi destinati agli utenti. Nonostante le nuove prospettive offerte da tale progresso in termini di possibilità e innovazione, emergono dei seri interrogativi. Ad esempio, la crescente complessità dei modelli di machine learning ha minato la fiducia dei consumatori nei confronti dei metodi e dei risultati ottenuti da tali sistemi. Gli algoritmi più efficienti vengono infatti spesso definiti "scatole nere" visto la difficoltà dell'utente di determinare come le loro previsioni vengano raggiunte. Inoltre, è possibile che in tali decisioni ci possano essere stati commessi errori dovuti a bias introdotte durante il processo di addestramento. La limitata interpretabilità di questi modelli rende arduo individuare e correggere eventuali malfunzionamenti. Di fronte a tali problematiche si potrebbe essere tentati al ricorrere ad algoritmi che garantiscano una più facile interpretazione vista la loro semplicità. Nonostante ciò, è importante sottolineare che tale decisione comporta frequentemente una diminuzione dell'accuratezza dei risultati, una circostanza spesso inaccettabile data l'importanza della precisione in determinati contesti in cui l'intelligenza artificiale viene impiegata. Come suggerito da Christoph Molnar [3], una possibile soluzione potrebbe risiedere nell'utilizzo di metodi di interpretazione *Model-Agnostic*. Strumenti come l'Explainable Artificial Intelligence (XAI) o Interpretable Machine Learning (IML) consentono, infatti, di spiegare qualsiasi modello, indipendentemente dalla sua complessità o grado di interpretabilità. Il processo alla base del funzionamento della maggior parte di questi metodi può essere riassunto nel SIPA *framework* [4]:

1. **S**AMPLING STAGE: si seleziona in maniera casuale un sottoinsieme del dataset
2. **I**NTERVENTION STAGE: si modificano i valori delle feature dei campioni estratti con l'intento di alterare le previsioni del modello
3. **P**REDICTION STAGE: si utilizza il modello sui nuovi dati per ottenere previsioni differenti da quelle trovate in precedenza
4. **A**GGREGATION: si osserva come le previsioni siano cambiate per capire il perché delle scelte fatte dal modello

Concludendo, l'obiettivo principale di questa tesi è quello di affrontare la crescente complessità dei modelli di machine learning, offrendo una soluzione mediante lo studio e l'applicazione pratica di SHAP (Shapley Additive exPlanations) uno dei metodi di interpretazione Model-Agnostic. Adottando le metodologie illustrate da Christoph Molnar nel suo libro "*Interpreting Machine Learning Models with SHAP*" [5], l'elaborato si propone di fornire una comprensione chiara e dettagliata di come SHAP possa essere impiegato come strumento interpretativo per spiegare le previsioni di una Support Vector Machine (SVM).

Capitolo 2

I Valori di Shapley

Nella teoria dei giochi cooperativa, i *valori di Shapley* costituiscono una soluzione per stabilire in modo equo la suddivisione di una *ricompensa* (il *payout*), prendendo come riferimento i contributi forniti dai partecipanti al gioco. In questo capitolo, partendo dall'esempio di un gioco elementare, sarà spiegata la teoria alla base dei valori di Shapley, illustrandone il loro calcolo e i relativi assiomi.

2.1 Il Panino al burro di arachidi e marmellata

Si consideri il seguente esempio concreto. Da qualche mese, Mario si allena in palestra e di recente ha iniziato a includere nella sua routine un panino con burro di arachidi e marmellata come merenda preallenamento. Tuttavia, Mario si domanda quanto ciascuno dei tre ingredienti contribuisca alla bontà dello spuntino. Quindi decide di provare diverse combinazioni, dando un voto a ciascuna di esse che tenga in conto del tempo di preparazione del panino, del suo costo totale e del gusto.

Il problema può essere visto come un gioco cooperativo dove i partecipanti sono i tre ingredienti: **TP** = Tostatura del Pane, **BA** = Burro di Arachidi e **M** = Marmellata; mentre la ricompensa è il voto dato da Mario. (In questa analisi si preferisce considerare come ingrediente la tostatura piuttosto che il pane, poiché in caso contrario per alcune combinazioni si avrebbero solo burro e/o marmellata, invece che un effettivo panino)

Tabella 2.1 - Combinazioni di ingredienti

Ingredienti	Voto	Note
{Ø}	0	Un panino vuoto, non tostato
{TP}	1	La tostatura del pane richiede tempo, ma migliora il panino
{BA}	3	Al pane non tostato, è aggiunto il burro di arachidi
{M}	4	Al pane non tostato, è aggiunta la marmellata
{TP, BA}	5	Il pane viene tostato, e è aggiunto il burro di arachidi
{TP, M}	4	Il pane viene tostato, e è aggiunta la marmellata
{BA, M}	8	Panino senza tostatura, ma con burro di arachidi e marmellata
{TP, BA, M}	10	Panino tostato a cui sono aggiunti burro di arachidi e marmellata

2.2 Contributi Marginali

La Tabella 2.1 fornisce una panoramica preliminare dei contributi di ciascun partecipante al voto espresso da Mario. Per una distribuzione più equa, è possibile utilizzare i dati raccolti per calcolare i contributi marginali degli ingredienti. In pratica, ciò implica calcolare la differenza tra il valore della coalizione che include il giocatore e quella che lo esclude. Ad esempio, confrontando il voto tra la coalizione {Tostatura Pane, Marmellata} e il voto del panino contenente solo la Marmellata, è possibile trovare il contributo marginale della Tostatura del Pane alla coalizione {Marmellata}. In questo esperimento si ottiene: 5 – 4 = 1. Viceversa, se si volesse determinare il contributo dato dall' aggiungere la marmellata al pane tostato si avrebbe un diverso risultato: 5 – 1 = 4. In modo analogo si possono calcolare tutti i contributi marginali.

Tabella 2.2 - Contributi Marginali

Ingrediente Aggiunto	Alla Coalizione	Voto Con Aggiunta	Voto Senza Ingrediente	Contributo Marginale
TP	{∅}	1	0	1
TP	{BA}	5	3	2
TP	{M}	4	4	0
TP	{BA, M}	10	8	2
BA	{∅}	3	0	3
BA	{TP}	5	1	4
BA	{M}	8	4	4
BA	{TP, M}	10	4	6
M	{∅}	4	0	4
M	{TP}	4	1	3
M	{BA}	8	3	5
M	{TP, BA}	10	5	5

Con l'ausilio della tabella 2.2 e tramite medie pesate dei loro contributi marginali, è possibile calcolare i valori di Shapley di ogni giocatore, trovando quindi in che misura ciascun partecipante abbia influenzato il sapore del panino.

Esistono diversi approcci per determinare dei pesi che assicurino una distribuzione equa del voto. Uno di questi consiste nell'esaminare tutte le possibili permutazioni degli ingredienti.

1. (TP, BA, M) 2. (TP, M, BA) 3. (BA, TP, M)
4. (M, TP, BA) 5. (BA, M, TP) 6. (M, BA, TP)

Per ogni giocatore, infatti, si considerando gli ingredienti che lo precedono nelle permutazioni. Questi sottoinsiemi formano le coalizioni a cui il giocatore è stato aggiunto e, ignorando l'ordine degli elementi, si conta il numero di volte che una coalizione è stata ripetuta. Tali conteggi possono quindi essere utilizzati come pesi dei contributi marginali.

Per esempio, si può vedere come la Tostatura del Pane sia stata aggiunta 2 volte (in 1. e in 2.) alla coalizione insieme vuoto {∅} (il panino senza niente) , 1 volta (in 3.) ad un panino con Burro di Arachidi, 1 volta (in 4.) ad un panino con Marmellata e 2 volte (in 5. e in 6.) ad un panino con Burro di Aachidi e Marmellata.

Trovati i pesi e usando i contributi marginali della tabella 2.2 si può calcolare il valore di Shapley della Tostatura del Pane nel seguente modo:

$$\phi_{TP} = \frac{1}{6}(2 \cdot 1 + 1 \cdot 2 + 1 \cdot 0 + 2 \cdot 2) = \frac{8}{6} \approx 1.34$$

Analogamente si ottengono i valori di Shapley del Burro di Arachidi e della Marmellata

$$\phi_{BA} = \frac{1}{6}(2 \cdot 3 + 1 \cdot 4 + 1 \cdot 4 + 2 \cdot 6) = \frac{26}{6} \approx 4.33$$

$$\phi_{M} = \frac{1}{6}(2 \cdot 4 + 1 \cdot 3 + 1 \cdot 5 + 2 \cdot 5) = \frac{26}{6} = \approx 4.33$$

Sommando i contributi individuali di ogni giocatore si può vedere come si ottenga il voto dato da Mario al panino completo: $1.34 + 4.33 + 4.33 = 10$. È interessante osservare come, nonostante il Burro di Arachidi abbia ricevuto da Mario un punteggio più basso rispetto alla Marmellata nella Tabella 2.1, entrambi gli ingredienti contribuiscano in modo equo al sapore complessivo del panino. Ciò è il risultato di una migliore sinergia tra il Burro di Arachidi e il Pane Tostato rispetto a quella di quest'ultimo ingrediente con la Marmellata.

2.3 La Formula dei Valori di Shapley

L'esempio dei paragrafi precedenti può essere esteso a un numero generico di partecipanti " n ". Sia $N = \{1, \dots, n\}$ l'insieme dei giocatori, e sia 2^N l'insieme di tutte le sue possibili coalizioni. Un *gioco cooperativo* è la coppia (N, v), formata da N e dalla funzione $v : 2^N \to \mathbb{R}$ chiamata *funzione valore* (o *funzione caratteristica*) che mappa le coalizioni $S \in 2^N$ ai loro rispettivi payout (ovvero i voti della Tabella 2.1). L'*insieme dei giochi operativi su N* si indica con $\mathcal{G}_N$. Si definisce con il termine di *valore* una mappa $\phi : \mathcal{G}_N \to \mathbb{R}^n$, che ad ogni gioco (N, v) associa un vettore $\phi(v)$ di componenti $\phi_j(v)$ con $j = 1, \dots, n$. In particolare, il *valore di Shapley del j-esimo giocatore* sono le componenti della mappa aventi la seguente formula:

$$\phi_j(v) = \sum_{S \subseteq N \setminus \{j\}} \frac{|S|!\,(n - |S| - 1)!}{n!} \big(v(S \cup \{j\}) - v(S)\big) \qquad (2.1)$$

Analogamente a quanto fatto nei paragrafi precedenti, l'idea è quella di determinare la media dei contributi marginali alla ricompensa. Quindi per ogni coalizione S non contenente il j-esimo giocatore si calcola il contributo marginale come differenza tra la ricompensa associata all'insieme contenente S e j, e quella della coalizione S stessa: $(v(S \cup \{j\}) - v(S))$

In modo simile a quanto descritto nel paragrafo 2.2, per attribuire un peso ad un contributo marginale, si esaminano le permutazioni di n elementi e si calcola il numero di volte in cui l'insieme S, indipendentemente dall'ordine dei suoi elementi, precede il giocatore j in una di queste permutazioni. Per trovare tale quantità basta considerare il fatto che dall'insieme di tutte le permutazioni, quelle che interessano sono formate da: " una permutazione dell'insieme S ", seguita " dall'elemento j ", e infine " una permutazione dei $n - |S| - 1$ giocatori restanti ". Ciò significa che i pesi dei contributi sono dati dal prodotto fra $|S|!$ il numero di permutazioni dell'insieme S, e $(n - |S| - 1)!$ il numero di permutazioni degli elementi $\notin$ alla coalizione. La formula prevede quindi la somma di tutti i prodotti fra contributi marginali delle varie coalizioni S e il loro rispettivo peso, ciascuno diviso per $n!$ il numero delle possibili permutazioni di n giocatori

2.4 Gli Assiomi alla base dei Valori di Shapley

La scelta di usare il valore di Shapley come "modo giusto" per dividere una ricompensa è giustificata dal suo soddisfare una serie di assiomi:

■ EFFICIENZA : la somma dei contributi di ogni giocatore deve essere uguale alla ricompensa

$$\sum_{j=1}^{n} \phi_j = v(N)$$

■ SIMMETRIA : se due giocatori *j* e *k* hanno gli stessi contributi marginali, allora essi devono contribuire in maniera equivalente alla ricompensa

$$v(S \cup \{j\}) = v(S \cup \{k\}) \,, \;\; \forall \; S \subseteq N \backslash \{j,k\} \implies \phi_j = \phi_k$$

■ GIOCATORE ININFLUENTE (O NULLO) : un giocatore il cui contributo marginale in ciascuna coalizione S è nullo, possiede un valore di Shapley uguale a zero

$$v(S \cup \{j\}) \; = \; v(S) \,, \;\; \forall \; S \subseteq N \backslash \{j\} \implies \phi_j = 0$$

■ ADDITIVITÀ : nel caso di un gioco in cui esistano due funzioni valori v_1 e v_2 si ha che il valore di Shapley della loro somma è uguale alla somma dei singoli valori di Shapley

$$\phi_j(v_1 + v_2) = \; \phi_j(v_1) + \phi_j(v_2)$$

Capitolo 3

Risoluzione di un Problema di Classificazione tramite SVM

Il terzo capitolo di questa tesi si distacca momentaneamente dalla trattazione teorica dei valori di Shapley per immergersi nell'analisi di un dataset e la risoluzione di un problema di classificazione, ad esso associato, mediante Support Vector Machines (SVM). Tale digressione è finalizzata a preparare il terreno per l'applicazione dei valori di Shapley nel capitolo successivo, mostrando la necessità dell'utilizzo di tali strumenti nell'interpretazione di modelli black-box.

3.1 Il Dataset Wine Quality

Il dataset analizzato, appartenente alla Machine Learning Repository UCI [6], contiene i risultati delle analisi relative a dei campioni di un vino rosso del Portogallo. Tali dati sono impiegati per lo studio della qualità di tali prodotti. Le variabili dei dataset, infatti, si dividono in 11 feature ottenute con dei test fisico-chimici e 1 feature qualitativa, basata su opinioni sensoriali di esperti nel settore.

Tabella 3.1 - Attributi Dataset

Attributi	Spiegazione Attributi
fixed acidity	gli acidi sono uno dei maggiori costituenti dei vini e contribuiscono notevolmente al loro sapore. Gli acidi si dividono in acidi fissi e volatili e risiedono maggiormente nei chicchi d'uva, ma si trasmettono anche al vino
volatile acidity	generalmente legati al deterioramento del vino. Alti livelli possono comportare un sapore negativo tendente all'aceto
citric acid	presente in piccole quantità, viene aggiunto ai vini per aumentarne l' acidità e donare freschezza.
residual sugar	residui rimasti dopo la fermentazione, derivati dagli zuccheri dell'uva
chlorides	sali minerali acidi che contribuiscono alla sapidità del vino
free sulfur dioxide	il biossido di zolfo SO_2 viene utilizzato per le sue proprietà antiossidanti. Al ph del vino l'SO_2 si trova in due forme: combinata e, in piccola parte libera
total sulfur dioxide	la somma tra la quantità di SO_2 libera e combinata
density	il rapporto $Massa/Volume$
ph	misura l' acidità/basicità del vino, in una scala che va da 0 a 14
suplhates	additivi che contribuiscono alla conservazione del vino
alcohol	la percentuale di alcol contenuta nel vino
quality	i vini sono classificati in una scala da 1 a 10 in base alla loro qualità

3.2 Problema di Classificazione

Come suggerito nella pagina web della repository UCI al data frame “winequality” può essere associato un problema di classificazione. I vari vini, infatti, vengono divisi in classi tramite la feature “quality”. L’ obiettivo sarebbe quindi quello di allenare un algoritmo di classificazione in modo da creare un modello capace di predire l’eccellenza di nuovi campioni di vino, attraverso i valori dei test fisico-chimici.

Il problema non è però semplice, un conteggio delle classi mostra che il dataset non è bilanciato, ci sono molti più vini aventi una qualità media rispetto a quelli con una cattiva o ottima qualità. Inoltre alcune classi (1, 2, 9) non sono associate a nessun campione.

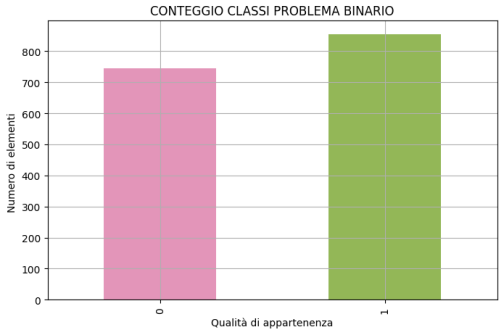
Qualità	Conteggi	Frequenze
3	10	0.006254
4	53	0.033146
5	681	0.425891
6	638	0.398999
7	199	0.124453
8	18	0.011257

Tabella 3.1 - Distribuzione Qualità

Figura 3.1 - Conteggio Classi

Il non bilanciamento delle classi del dataset rende difficile l’applicazione di metodi di classificazione, soprattutto considerando anche il poco numero di campioni a disposizione.

Visto che l’obiettivo principale della tesi è quello di mostrare come i valori di Shapley possano essere usati per interpretare un modello, è stato deciso di semplificare il problema dividendo i vari campioni di vino in due macro-classi (CLASSE 0: i vini sotto la media: qualità da 1 a 5, CLASSE 1: i vini sopra la media: qualità da 6 a 10). Ricontando le classi si può vedere come il dataset risulti più bilanciato.



3.2.1 Support Vector Machine

Le *Macchine a Vettori di Supporto* (SVM) 30[7] sono dei modelli di apprendimento supervisionato associati ad algoritmi che analizzano dati con l'obiettivo di classificarli. Dato un *training set* $T = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\} \subset \mathbb{R}^n \times \{\pm 1\}$ un algoritmo di addestramento per SVM definisce un modello $f: \mathbb{R}^n \to \{+1, -1\}$, che, dopo un allenamento su T, è in grado di assegnare a nuovi esempi una delle due classi di appartenenza, eseguendo quindi una *classificazione binaria*.

Si indichi con $\pi_{\mathrm{w,b}} \coloneqq \{x \in \mathbb{R}^n | w^T x + b = 0\}$ l'*iperpiano* di $\mathbb{R}^n$ definito dal vettore normale $w \in \mathbb{R}^n$ e dal parametro $\mathrm{b} \in \mathbb{R}$, (con $w^T x$ che indica il prodotto scalare $< w, x >$). In particolare, sia $dist(\pi_{\mathrm{w,b}}, x) = \frac{|w^T x + b|}{||\mathrm{w}||}$ la distanza euclidea tra un vettore $x \in \mathbb{R}^n$ e $\pi_{\mathrm{w,b}}$. L'obiettivo delle SVM è quello di trovare l'iperpiano $\pi_{w,b}$ che meglio divide i vettori $\{x^{(1)}, \dots, x^{(n)}\}$ nelle due classi di appartenenza ±1, massimizzando cioè il *margine di separazione* $r = dist(\pi_{\mathrm{w,b}}, x^{(a)})$, (dove $x^{(a)}$ è il punto del dataset più vicino all' iperpiano). Si vorrebbe quindi che gli esempi di T si trovino ad una distanza dall'iperpiano che sia maggiore di r (nella rispettiva direzione a seconda che essi siano positivi o negativi). Ciò si traduce con la seguente disuguaglianza: $y^{(i)}(w^T x^{(i)} + b) \geq r$

Imponendo come ipotesi che il vettore w abbia norma euclidea unitaria, ($||w||_2 = 1$), si ottiene un problema di ottimizzazione vincolato:

$$\begin{cases} \max\limits_{\mathrm{w,b,r}} r \\ \text{subject to } y^{(i)}(w^T x^{(i)} + b) \geq r\,, \forall i = 1, \dots, m\,, ||w||_2 = 1\,, r > 0 \end{cases}$$

Senza perdere di generalità si può scegliere una scala di misurazione per i dati in modo che il valore del predittore $w^T x + b$ sia uguale a 1 per il campione più vicino all'iperpiano (cioè per $x^{(a)}$). Sfruttando inoltre la bilinearità del prodotto scalare si dimostra che sotto queste ipotesi $r = \frac{1}{||w||}$. Lasciando variare $||w||$ ci si limita quindi a chiedere che i punti si trovino almeno a distanza 1 iperpiano, invece che a distanza r, ovvero si usa la disequazione: $y^{(i)}(w^T x^{(i)} + b) \geq 1$

Considerando infine che massimizzare $r = \frac{1}{||w||}$ è equivalente a minimizzare $\frac{1}{2}||w||^2 = \sum_{j=1}^{n} w_j^{\mathbf{2}}$ il problema per trovare l' iperpiano separatore ottimale può essere così riscritto:

$$\textbf{HARD MARGIN SVM: } \begin{cases} \min\limits_{w,b} \frac{1}{2}||w||^2 \\ \text{subject to } y^{(i)}(w^T x^{(i)} + b) \geq 1\,, \forall i = 1, \dots, m \end{cases}$$

3.2.2 Il Problema Duale

Il problema Hard Margin SVM può essere risolto in maniera più efficiente passando alla sua *formulazione duale*. Si introduce cioè una *variabile di slack* $\alpha^{(i)}$, che indica quanto il vincolo sia importante per la soluzione, e si calcola la *lagrangiana* del problema:

$$\begin{cases} L(w, \alpha) = \frac{1}{2} \sum_{j=1}^{n} w_{\mathrm{j}}{}^2 - \sum_{i=1}^{m} \alpha^{(i)}(y^{(i)} w^T x^{(i)} - 1) \\ s.t\ \alpha_i \geq 0, \forall i = 1, \dots, m \end{cases}$$

Bisogna quindi minimizzare su *w* e massimizzare su α ottenendo:

$$\begin{cases} \max_{\alpha} J(\alpha) = \sum_{i=1}^{m} \alpha^{(i)} - \frac{1}{2} \sum_{i=1}^{m} \sum_{k=1}^{m} \alpha^{(i)} \alpha^{(k)} y^{(i)} y^{(k)} x^{(i)^T} x^{(k)} \\ s.t\ \alpha^{(i)} \geq 0, \forall i = 1, \ldots, m, \quad \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} = 0 \end{cases}$$

Si indichi con *SV* (*Vettore di Supporto*) l'insieme degli indici dei campioni più vicini all'iperpiano (possono essere più di uno,) si ha quindi che la *funzione di decisione* è data da:

$$f(x) = sign(w^T x) = sign(\sum_{i \in SV} \alpha^{(i)} y^{(i)} x^T x^{(i)} + b$$

$$dove\ b = \frac{1}{|SV|} \sum_{i \in SV} (y^{(i)} - \sum_{k \in SV} \alpha^{(k)} y^{(k)} x^{(i)^T} x^{(k)} + b)$$

Risolvendo il problema della lagrangiana si vanno a trovare gli $\alpha^{(i)}$ da inserire in J(α) e facendo così si hanno due casi possibili:

- CASO $\alpha^{(i)} > 0$: risulta che $y^{(i)}(w^{\mathrm{T}} x^{(i)}) = 1)$ e quindi che il punto $x^{(i)}$ è un vettore di supporto
- CASO $\alpha^{(i)} = 0$: il punto $x^{(i)}$ non è un vettore di supporto

Data la *soluzione ottimale* α^* del problema duale si ha che i pesi ottimali sono:

$$w^* = \sum_{i \in SV} \alpha^{*(i)} y^{(i)} x^{(i)} \qquad b^* = \frac{1}{|\mathrm{SV}|} \sum_{i \in \mathrm{SV}} (y^{(i)} - w^{*T} x^{(i)})$$

3.2.3 Soft Margin e Metodi Kernel

Non sempre i campioni di un dataset sono linearmente separabili, è quindi necessario in questi casi introdurre una tolleranza sulla possibilità di oltrepassare il margine per i vettori $x^{(i)}$. Si introduce quindi una *variabile di slack* $\xi^{(i)}$, associata al campione $(x^{(i)}, y^{(i)})$, che permette al campione in questione di essere all'interno del margine o dal lato sbagliato dell'iperpiano. Si procede quindi a sottrarre il valore di $\xi^{(i)}$ dal margine, imponendo che $\xi^{(i)} \geq 0$

Il problema può essere quindi riscritto nel seguente modo:

SOFT MARGIN SVM: $\begin{cases} \min_{w} \frac{1}{2} ||w||^2 + C \sum_{i=1}^{m} \xi^{(i)} \\ \text{subject to } y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi^{(i)},\ \xi^{(i)} \geq 0,\ \forall i = 1, \ldots, m \end{cases}$

Il parametro $C \in \mathbb{R}^+$, che prende il nome di *regolarizzazione*, modifica la dimensione del margine e il quantitativo di slack totale (tanto più è piccolo è tanto più l'SVM sarà morbida). Un altro accorgimento che si può fare quando il dataset non è linearmente separabile consiste nel proiettare i campioni del dataset su uno spazio a dimensione maggiore.

Sia X l' insieme dei campioni del dataset, si costruisce una mappa $\phi: X \subset \mathbb{R}^n \to \hat{X} \subset \mathbb{R}^N$ che collega lo spazio di input con lo spazio di arrivo. Una volta trovato il piano separatore tramite SVM, si procede con il farne la controimmagine tornando nella dimensione iniziale.

Se i punti del dataset sono difficili da separare, può succedere che la dimensione del feature space sia estremamente grande e non si può quindi usare $\phi(x^{(i)})$ per calcolare la $J(\alpha)$ della SVM. In questi casi si trova un *Kernel* K tale che $K(x^{(i)}, x^{(k)}) = < \phi(x^{(i)}), \phi(x^{(k)}) >$, riuscendo quindi ad evitare di dover conoscere ϕ esplicitamente. Ci sono vari tipi di kernel:

■ KERNEL LINEARE: $K(x^{(i)}, x^{(k)}) = < x^{(i)}, x^{(k)} >$

■ KERNEL POLINOMIALE: $K(x^{(i)}, x^{(k)}) = < x^{(i)}, x^{(k)} >^d$

■ KERNEL GAUSSIANO (o RBF): $K(x^{(i)}, x^{(k)}) = exp\left(-\frac{1}{2\sigma^2}\left||x^{(i)} - x^{(k)}\right||_2^2\right)$

■ KERNEL SIGMOIDALE: $K(x^{(i)}, x^{(k)}) = \tanh(< x^{(i)}, x^{(k)} >)$

3.3 Ottimizzazione degli Iperparametri

Quando si allena una SVM, è possibile specificare diversi parametri che influenzano il modello e ne alterano le capacità predittive. L'obiettivo è individuare, durante la fase di addestramento del classificatore, i migliori parametri adatti ai dati in esame. Si procede quindi alla creazione di k modelli diversi $\hat{f}_1, \dots, \hat{f}_k$, ognuno caratterizzato da diverse combinazioni di iperparametri al fine di individuare quello che produce i risultati migliori. In particolare, considerando un dataset D composto dalle coppie features-target $\{(x^{(1)}, y^{(1)}), \dots, (x^{(d)}, y^{(d)})\} \subset \mathbb{R}^d \times \mathbb{R}$, si effettua la seguente divisione di D:

■ TRAINING SET (T): è l'insieme su cui si addestrano i vari modelli $\hat{f}_1, \dots, \hat{f}_k$

■ VALIDATION SET (V): è l'insieme su cui si testano i k modelli e si identifica il migliore.

■ TEST SET (P): è l'insieme su cui si analizzano le performance del modello migliore trovato, in modo da capire se possa essere usato per fare previsioni.

Per valutare in maniera oggettiva i risultati dei classificatori si utilizzano una serie di metriche:

■ ACCURACY: l'*accuratezza* su un Test Set (P) è il rapporto tra le previsioni corrette del classificatore su P e il numero di predizioni possibili:

$$Acc(P) = \frac{num.\ pred.\ corrette\ su\ P}{cardinalità\ di\ P}$$

L'accuratezza non è sempre un indicatore esaustivo. Questo è particolarmente vero nei casi in cui il dataset abbia classi non bilanciate.

■ MATRICE DI CONFUSIONE (A): la *matrice di confusione* rispetto a P è la matrice che ha come elemento $a_{i,j}$ il numero di campioni appartenenti alla classe C_i che il modello ha predetto di appartenere alla classe C_j. Sulla diagonale principale della matrice si potranno vedere le previsione corrette dell'algoritmo, mentre gli altri elementi indicheranno i numeri di errori fatti. Si useranno tre matrici di confusione.

Sia, infatti, $a_{i,j}$ definito come sopra e sia 'c' il numero di classi presenti nel dataset, allora si definiscono le matrici aventi i seguenti elementi:

1. MATRICE DI CONFUSIONE NON NORMALIZZATA:

$$A_{i,j} = a_{i,j}$$

2. MATRICE DI CONFUSIONE NORMALIZZATA RISPETTO LE VERE CLASSI:

$$A_{i,j} = \frac{a_{i,j}}{(a_{i,0} + a_{i,1} + \ldots + a_{i,j} + \ldots + a_{i,c})}$$

3. MATRICE DI CONFUSIONE NORMALIZZATA RISPETTO LE CLASSI PREDETTE:

$$A_{i,j} = \frac{a_{i,j}}{(a_{0,i} + a_{1,i} + \ldots + a_{j,i} + \ldots + a_{c,i})}$$

■ PRECISION: la *precision* è l'abilità del classificatore di non etichettare come appartenente alla classe C_i un elemento che non le appartiene

$$\text{prec}(C_i, P) = \frac{num.\ di\ elem.\ correttamente\ predetti\ \in C_i}{num.\ totale\ di\ elem.\ predetti\ \in C_i} = \frac{a_{i,i}}{somma\ elem.\ in\ col.\ i-esima}$$

■ RECALL: la *recall* è l'abilità del classificatore di trovare tutti gli elementi appartenenti alla classe C_i

$$\text{rec}(C_i, P) = \frac{num.\ di\ elem.\ correttamente\ predetti\ \in C_i}{num.\ totale\ di\ elem.\ \in C_i} = \frac{a_{i,i}}{somma\ elem.\ in\ riga\ i-esima}$$

Sia per la precision che per la recall si useranno i loro valori medi

■ F - BETA SCORE: il *F-beta Score* è la media pesata della Precision e la Recall, assume sempre valori compresi fra 0 e 1 (0 minimo, 1 massimo), ed è un ulteriore parametro per misurare l' accuratezza del modello

$$F_\beta(C_i, P) = (1 + \beta^2)\frac{\text{prec}(C_i, P) \cdot \text{rec}(C_i, P)}{\beta^2 \text{prec}(C_i, P) + \text{rec}(C_i, P)}$$

In questo lavoro è stato utilizzato l' F_1 Score

3.3.1 La GridSearchCV

La *Grid Search Cross-Validation* è una funzione della libreria *Sklearn*, che mediante un approccio sistematico consente di esplorare tutte le combinazioni possibili di parametri prese da una griglia predefinita. Il termine "Cross-Validation" si riferisce alla suddivisione del dataset in vari sottoinsiemi, permettendo di addestrare il modello in modo da evitare risultati influenzati dalla distribuzione specifica di alcuni dati. In questo lavoro sono stati usati i seguenti iperparametri:

■ PARAMETRO C: l'iperparametro C è la regolarizzazione del modello e permette di specificare il grado di durezza del margine della SVM. Alla Grid Search viene passata una lista contenente valori di diverso ordine di grandezza.

■ KERNEL: una lista contenente i nomi dei Kernel presentati a pag 12

■ PARAMETRO GAMMA γ: l'iperparametro gamma viene utilizzato solo dalle SVM con il Kernel Gaussiano. Tale valore permette di regolare il parametro σ^2. Infatti, si ha che $\gamma = \frac{1}{2\sigma^2}$

Prima di procedere con la Grid Search è necessario considerare un aspetto importante. Nelle SVM l'ottimale iperpiano che separa le classi è influenzato dalle dimensioni delle feature di input. Di conseguenza, è consigliabile standardizzare i dati prima di addestrare il modello. In condizioni normali, questa operazione sarebbe semplice, in quanto è sufficiente applicare la normalizzazione ai dati prima di passarli all'algoritmo. Tuttavia, durante l'utilizzo della Grid Search, la suddivisione del dataset avviene internamente, rendendo impossibile applicare direttamente tale pre-processing. Per aggirare il problema si può ricorrere allo strumento *Pipeline* di Sklearn. Le pipeline consentono di concatenare in modo sequenziale un numero arbitrario di passaggi di elaborazione dei dati, così che l'output di un passo diventi l'input del successivo. Pertanto, è possibile creare una Pipeline che includa inizialmente uno StandardScaler e successivamente la SVM. Pipeline che può quindi essere utilizzata nella Grid Search, garantendo così una corretta standardizzazione dei dati durante la ricerca dei migliori parametri del modello.

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_svc__C	param_svc__gamma	param_svc__kernel	params	split0_test_score	mean_test_score	std_test_score	rank_test_score
40	0.069996	0.0	0.020917	0.0	2	0.090909	rbf	{'svc__C': 2, 'svc__gamma': 0.0909090909090909...	0.772658	0.772658	0.0	1

Figura 3.3 - Best Estimator

Delle 60 SVM che sono state allenate per questo lavoro, quella che ha performato più delle altre è quella avente kernel Gaussiano, parametro $C = 2$ e parametro $\gamma = 0.09$. Una volta ottenuti i risultati della Grid Search, il miglior modello viene riaddestrato sul Training Set, per poi essere messo alla prova su Validation e Test Set. A seguire si riportano le metriche dell'SVM dopo l'allenamento:

	Accuracy	**Precision**	**Recall**	**F1**
Training	0.828810	0.830581	0.828810	0.828988
Validation	0.771875	0.784453	0.771875	0.772658
Test	0.746250	0.746923	0.746250	0.746389

Tabella 3.2 - Metriche Best Estimator

	classe 0	**classe 1**
classe 0	285	95
classe 1	108	312

Tabella 3.3 - Matrice di Confusione Non Normalizzata

	classe 0	**classe 1**
classe 0	0.750000	0.250000
classe 1	0.257143	0.742857

Tabella 3.4 - Matrice di Confusione Normalizzata Rispetto le Vere Classi

	classe 0	**classe 1**
classe 0	0.725191	0.233415
classe 1	0.274809	0.766585

Tabella 3.4 - Matrice di Confusione Normalizzata Rispetto le Classi Predette

L'SVM ha ottenuto dei discreti risultati. Le metriche sono bilanciate e non si sono verificati fenomeni di overfitting. Osservando le tabelle 3.3 e 3.4 si può vedere come il modello sia leggermente migliore nel riconoscere campioni di vino con qualità inferiore alla media.

Capitolo 4

Dai Valori di Shapley a SHAP

Nel secondo capitolo, sono stati trattati i valori di Shapley da un punto di vista della teoria dei giochi cooperativa, illustrando il loro impiego nella formulazione di una equa divisione di ricompense. Questo capitolo espone l'applicazione di tali concetti nell'ambito del machine learning, mostrando il loro utilizzo per un'analisi delle previsioni generate da un modello di apprendimento.

4.1 I Valori di SHAP

Una previsione può essere vista come un gioco cooperativo, in cui ciascuna variabile di un campione è trattata come un partecipante, e il valore generato dal modello è la ricompensa.

Si consideri un dataset formato da n feature, che contiene dati generati da una variabile casuale multivariata $X = (X_1, \dots, X_n)$ e su cui viene allenato un modello f. Dato un campione $x^{(i)}$ del dataset e una coalizione di variabili $S \subseteq N$, indicando con $C = N \backslash S$ l'insieme delle feature non appartenenti ad S, si può definire la seguente funzione caratteristica:

$$v_{f,x^{(i)}}(S) = \int f(x_S^{(i)} \cup X_C)\, d\mathbb{P}_{X_C} - E_X(f(X))$$

dove $x_S^{(i)} \cup X_C$ è un vettore di feature $\in \mathbb{R}^p$ che contiene i valori di $x^{(i)}$ nelle posizioni in S e per le restanti posizioni in C è formato da dati generati da X_C , ovvero il vettore casuale contenente le variabili marginali di X negli indici corrispondenti. Ciò significa che è sufficiente passare come input solo la parte dei valori "*conosciuti*" di $x^{(i)}$, (quelli indicizzati in S) mentre le restanti feature sono trattate come variabili casuali, e la funzione val sarà in grado di restituire la previsione del modello f. La seconda parte dell'equazione $E_X(f(X))$ è aggiunta per assicurare che il valore della coalizione vuota $val(\emptyset)$ sia 0.

Dopo aver definito la funzione valore è possibile calcolare il contributo marginale di una variabile j ad una coalizione di feature S:

$$v(S \cup \{j\}) - v(S) = \int f\left(x_{S\cup\{j\}}^{(i)} \cup X_{C\backslash\{j\}}\right) d\mathbb{P}_{X_{C\backslash\{j\}}} - E_X(f(X)) - \left(\int f(x_S^{(i)} \cup X_C)\, d\mathbb{P}_{X_C} - E_X(f(X))\right) =$$
$$= \int f\left(x_{S\cup\{j\}}^{(i)} \cup X_{C\backslash\{j\}}\right) d\mathbb{P}_{X_{C\backslash\{j\}}} + \int f(x_S^{(i)} \cup X_C)\, d\mathbb{P}_{X_C}$$

Tale differenza mostra di quanto la previsione cambi quando si conosce anche la feature j oltre che quelle nella coalizione S.

Sostituendo il contributo marginale nell'equazione (2.1) si ottiene il valore di Shapley per j:

$$\phi_j^{(i)}(v_{f,x^{(i)}}) = \sum_{S \subseteq N \setminus \{j\}} \frac{|S|!\,(n-|S|-1)!}{n!} \left(\int f\left(x_{S\cup\{j\}}^{(i)} \cup X_{C\setminus\{j\}}\right) d\mathbb{P}_{X_{C\setminus\{j\}}} + \int f\left(x_S^{(i)} \cup X_C\right) d\mathbb{P}_{X_C} \right) \quad (4.1)$$

Questa versione dei valori di Shapley adattata alla machine learning prende il nome di " SHAP ".

4.1.1 Assiomi di SHAP

I valori di SHAP soddisfano gli stessi assiomi alla base dei valori di Shapley:

■ EFFICIENZA : la somma dei valori di SHAP di tutte le feature deve essere uguale alla differenza fra la previsione del modello f per $x^{(i)}$ e la media di tutte le previsioni:

$$\sum_{j=1}^{n} \phi_j^{(i)} = f(x^{(i)}) - E_X\big(f(X)\big)$$

■ SIMMETRIA : se due variabili j e k hanno gli stessi contributi marginali, allora essi devono contribuire in maniera equivalente alla previsione:

$$v_{f,x^{(i)}}(S \cup \{j\}) = v_{f,x^{(i)}}(S \cup \{k\}),\ \forall\, S \subseteq N\setminus\{j,k\} \implies \phi_j^{(i)} = \phi_k^{(i)}$$

Questa proprietà è importante per garantire che l'attribuzione della funzione valore sia uguale a prescindere da come le feature sono ordinate nel dataset.

■ GIOCATORE ININFLUENTE (O NULLO) : un giocatore il cui contributo marginale in ciascuna coalizione S è nullo, possiede un valore di Shapley uguale a zero:

$$v_{f,x^{(i)}}(S \cup \{j\}) = v_{f,x^{(i)}}(S),\ \forall\, S \subseteq N\setminus\{j\} \implies \phi_j^{(i)} = 0$$

■ ADDITIVITÀ : nel caso di un gioco in cui esistano due funzioni utilità v_1 e v_2 si ha che il valore di SHAP della loro somma è uguale alla somma dei singoli valori di SHAP:

$$\phi_j^{(i)}(v_1 + v_2) = \phi_j^{(i)}(v_1) + \phi_j^{(i)}(v_2)$$

In alcuni modelli, come nelle foreste casuali, la previsione è media delle previsioni di più alberi. Quindi grazie all'additività, è possibile calcolare i valori di SHAP dei singoli modelli e sommarli.

.

4.2 Stima dei Valori di SHAP

Nel precedente paragrafo è stata mostrata la teoria alla base dei valori di SHAP. Il problema è che la formula (4.1) non è sempre utilizzabile dato che spesso non si ha una espressione chiusa di f oppure non si è a conoscenza della distribuzione X_C. In questi casi si procede con l'effettuarne una stima.

4.2.1 Metodo Monte Carlo

L'integrazione Monte Carlo rappresenta un metodo per stimare l'integrale di una funzione che dipende da una variabile casuale. Questo approccio prevede l'estrazione casuale di un insieme di campioni generati dalla variabile in questione, seguita dal calcolo della media dei valori della funzione quando tali campioni vengono utilizzati come input. Grazie a tale metodo è possibile stimare la funzione valore di SHAP campionando dal dataset i valori delle feature assenti da una coalizione (le X_C). La parte del dataset che viene usata prende il nome di "background data".

Si consideri un background di m esempi, il valore di una coalizione di S feature è così stimato:

$$\hat{v}(S) = \frac{1}{m}\sum_{k=1}^{m}\left(f(x_S^{(i)} \cup x_C^{(k)}) - f(x^{(k)})\right)$$

Il contributo marginale di una feature j è calcolato con una sommatoria invece che con un integrale:

$$\widehat{\Delta}_{S,j} = \hat{v}(S \cup \{j\}) - \hat{v}(S) = \frac{1}{m}\sum_{k=1}^{m}\left(f(x_{S\cup\{j\}}^{(i)} \cup x_{C\setminus\{j\}}^{(k)}) - f(x_S^{(i)} \cup x_C^{(k)})\right)$$

La formula di SHAP diventa quindi:

$$\hat{\phi}_j^{(i)} = \sum_{S\subseteq N\setminus\{j\}} \frac{|S|!\,(n-|S|-1)!}{n!}\,\widehat{\Delta}_{S,j}$$

4.3 Metodi di Approssimazione

Il costo computazionale del calcolo di SHAP cresce in modo esponenziale con il numero di feature n, visto il dover gestire 2^n coalizioni. Per n elevati, è conveniente utilizzare tecniche di approssimazione anziché considerare tutte le possibili coalizioni. Esistono diversi metodi di approssimazione, ciascuno contraddistinto da un differente equilibrio tra velocità ed accuratezza. Alcuni sono di natura generica, mentre altri sono ottimizzati per algoritmi specifici di machine learning. Questo studio si concentra sui metodi *model-agnostic*, utilizzabili indipendentemente dal modello considerato.

4.3.1 Kernel SHAP

Kernel SHAP fu la prima implementazione di un algoritmo per la stima dei valori di SHAP. Questo metodo si divide in cinque passi, che comprendono il campionamento delle coalizioni e l'esecuzione di una regressione lineare pesata:

1. Lanciando n volte una moneta, si genera una coalizione casuale $z' \in \{0,1\}^n$, ovvero un vettore in cui si usa il valore 1 per indicare la presenza di una feature e il valore 0 in caso contrario. Tale step viene ripetuto K volte, generando diverse coalizioni che insieme formano il dataset Z che verrà usato per l'allenamento del modello di regressione.
2. Ogni z' è convertita nello spazio delle feature utilizzando una funzione $h_x: \{0,1\}^n \to \mathbb{R}^n$, che mappa gli 1 con i valori delle corrispondenti variabili del campione $x^{(i)}$ che si sta spiegando e gli 0 con i valori delle variabili di un altro campione casualmente estratto dal dataset. Una volta ottenuto $h_x(z') = z$, utilizzando il modello f si calcola la previsione $f(z)$. Tali previsioni vengono usate dal modello di regressione come target.
3. Si determina un peso per ogni z' utilizzando lo SHAP kernel:
$$\pi_x(z') = \frac{(n-1)}{\binom{n}{|z'|}\,|z'|(n-z')}$$
4. Con i dati, i target e i pesi degli step 1,2 e 3, si può costruire il modello di regressione lineare:
$$g(z') = \phi_0 + \sum_{j=1}^{n}\phi_j^{(i)} z_j'$$
Il modello g viene allenato ottimizzando la seguente funzione di perdita:
$$L(f,g,\pi_x) = \phi_0 + \sum_{z'\in Z}[f(h_x(z')) - \mathrm{g}(\mathrm{z}')]^2\pi_x(z')$$
5. Si forniscono in output i valori di SHAP $\phi_j^{(i)}$, che corrispondono ai coefficienti del modello g

4.3.2 Approssimazione tramite Permutazioni

La stima attraverso permutazioni si fonda sulla selezione casuale di una permutazione delle feature, seguita dall'esecuzione di un processo di generazione in avanti e all'indietro di coalizioni. Si consideri per esempio un dataset con quattro variabili e sia $\sigma(1) = (x_2^{(i)}, x_3^{(i)}, x_1^{(i)}, x_4^{(i)})$ la prima permutazione casuale di un suo campione $x^{(i)}$. Nella generazione in avanti si parte da sinistra aggiungendo un elemento alla volta e si calcolano i contributi marginali delle coalizioni che si formano:

- $x_2^{(i)}$ aggiunto a $\{\emptyset\} \rightarrow$ si ottiene la coalizione $\{x_2^{(i)}\}$
- $x_3^{(i)}$ aggiunto a $\{x_2^{(i)}\} \rightarrow$ si ottiene la coalizione $\{x_2^{(i)}, x_3^{(i)}\}$
- $x_1^{(i)}$ aggiunto a $\{x_2^{(i)}, x_3^{(i)}\} \rightarrow$ si ottiene la coalizione $\{x_2^{(i)}, x_3^{(i)}, x_1^{(i)}\}$
- $x_4^{(i)}$ aggiunto a $\{x_2^{(i)}, x_3^{(i)}, x_1^{(i)}\} \rightarrow$ si ottiene la coalizione $\{x_2^{(i)}, x_3^{(i)}, x_1^{(i)}, x_4^{(i)}\}$

Ovviamente nelle coalizioni l'ordine degli elementi è ininfluente, qui viene preservato per una più comprensibile spiegazione. Si continua con la generazione all'indietro, dove si parte invece da destra:

- $x_4^{(i)}$ aggiunto a $\{\emptyset\} \rightarrow$ si ottiene la coalizione $\{x_4^{(i)}\}$
- $x_1^{(i)}$ aggiunto a $\{x_4^{(i)}\} \rightarrow$ si ottiene la coalizione $\{x_4^{(i)}, x_1^{(i)}\}$
- $x_3^{(i)}$ aggiunto a $\{x_4^{(i)}, x_1^{(i)}\} \rightarrow$ si ottiene la coalizione $\{x_4^{(i)}, x_1^{(i)}, x_3^{(i)}\}$
- $x_2^{(i)}$ aggiunto a $\{x_4^{(i)}, x_1^{(i)}, x_3^{(i)}\} \rightarrow$ si ottiene la coalizione $\{x_4^{(i)}, x_1^{(i)}, x_3^{(i)}, x_2^{(i)}\}$

La velocità di tale metodo deriva dal fatto che ad ogni passo-k, si calcolano due contributi marginali (uno per la generazione in avanti, uno per quella all'indietro) per ciascuna delle feature, invece che concentrarsi su una singola variabile. In questo modo, infatti, si possono sfruttare i valori della funzione valore calcolati dalla coalizione precedente per i contributi marginali della feature successiva, evitando così di chiamare più volte il modello. L'idea è quella di ripetere tale procedimento con "*p*" diverse permutazioni, ottenendo un numero sufficiente di contributi marginali che permetta, usando il metodo di Monte Carlo, di calcolare i valori di SHAP di ogni feature:

$$\hat{\phi}_j^{(i)} = \frac{1}{2p} \sum_{k=1}^{p} \left(\hat{\Delta}_{+\sigma(k),j} - \hat{\Delta}_{-\sigma(k),j} \right)$$

Per ogni permutazione casuale $\sigma(k)$, ci sarà stato un momento, in entrambe le generazioni, in cui la j-esima feature viene aggiunta ad una coalizione e vengono calcolati i contributi marginali $\hat{\Delta}_{\pm\sigma(k),j}$ Tali contributi vengono poi sommati alla fine per ottenere $\hat{\phi}_j^{(i)}$. Quindi, diversamente dalla formula teorica, in cui si somma per coalizioni e dove i contributi vengono pesati, si decide di sommare per permutazioni.

La precisione del metodo dipende dal numero di permutazioni casuali generate. Quando il numero di permutazioni è pari a $p = n!$, si riottiene la formula esatta, in quanto vengono utilizzate tutte le possibili combinazioni. Tuttavia, di norma, non è necessario iterare un numero così elevato di volte per ottenere risultati accurati. Di solito dieci permutazioni sono sufficienti. Nel corso degli anni, l'approccio di stima basato su permutazioni ha progressivamente sostituito il metodo Kernel per l'interpretazione agnostica dei modelli, grazie alla sua capacità di ottenere risultati accurati in una frazione del tempo richiesto dalla sua controparte. Pertanto, in questo lavoro, si è scelto di non impiegare il metodo Kernel per l'analisi della SVM implementata nel capitolo precedente, limitandosi a riportarlo per la sua rilevanza storica.

4.3.3 I valori di Owen e l'Approssimazione tramite Partizioni

Nei metodi precedenti si è sempre cercato di calcolare i valori di SHAP per singole feature, con l'obiettivo di determinare per ogni variabile, il suo contributo individuale al payout. In alcuni casi, potrebbe essere preferibile concentrarsi su un insieme di feature, che rappresentino un concetto simile. Per esempio, nel capitolo 2, Mario avrebbe potuto raggruppare il burro di arachidi e la marmellata in un'unica categoria di condimenti, cercando quindi di valutare quanto il loro contributo al sapore fosse più o meno rilevante rispetto alla tostatura del pane.

La stima per partizioni si basa sulla definizione di una gerarchia fra le feature, le quali vengono suddivise in gruppi in base alle loro correlazioni, seguendo una struttura ad albero. Partendo dal livello più alto si calcolano i valori di SHAP dei vari gruppi, trattando le variabili che li compongono come un'unica entità. Tali valori vengono poi attribuiti alle singole feature o ai sottogruppi del livello successivo. Questo processo ricorsivo è un esempio di gioco con una struttura di coalizione. I risultati ottenuti sono simili ai valori SHAP e prendono il nome di *valori di Owen 30*[8] [9]

Dato $N = \{1, \dots, n\}$, si indica con $\mathcal{T}(N)$ l'insieme delle partizioni di N, ovvero la collezione di tutti i modi in cui N può essere diviso in sottoinsiemi. Nella teoria dei giochi una $\mathcal{T} = \{T_1, \dots, T_m\} \in \mathcal{T}(N)$, prende il nome di *struttura di coalizione*, mentre i suoi elementi sono chiamati *unioni*. I *giochi cooperativi con strutture di coalizioni* sono le triple $(N, v, \mathcal{T})$. Il loro insieme è: $\mathcal{G}_N^{cs} = \mathcal{G}_N \times \mathcal{T}(N)$

Data la partizione $\mathcal{T} = \{T_1, \dots, T_m\}$ si può definire il *gioco quoziente* (M, u), avente come partecipanti i giocatori dell'insieme $M = \{1, \dots, m\}$ e funzione $v^{\mathcal{T}} := v/\mathcal{T}$ che $\forall\, H \subseteq M$ è data da:

$$v^{\mathcal{T}}(H) = v(Q) = v\left(\bigcup_{h \in H} T_h\right)$$

Si definisce con *valore coalizionale* la mappa $\Omega : \mathcal{G}_N^{cs} \to \mathbb{R}^n$ che ad ogni gioco con struttura $(N, v, \mathcal{T})$ associa un vettore $\Omega(v, \mathcal{T})$ di componenti $\Omega_j(v, \mathcal{T})$ con $j = 1, \dots, n$. Se ϕ è un valore su $\mathcal{G}_N$ allora si dice che Ω *è un valore coalizionale di* ϕ se e solo se $\Omega(v, \mathcal{T}) = \phi(v)$, $\forall\, v \in \mathcal{G}_N$.

I *valori di Owen del j-esimo giocatore* sono le componenti della mappa avente la seguente formula:

$$\Omega_j(v, \mathcal{T}) = \sum_{H \subseteq M \setminus \{k\}} \sum_{S \subseteq T_k \setminus \{j\}} \frac{|H|!\,(m - |H| - 1)!}{m!} \frac{|S|!\,(|T_k| - |S| - 1)!}{|T_k|!} \big(v(Q \cup S \cup \{j\}) - v(Q \cup S)\big)$$

dove $T_k \in \mathcal{T}$ è k l'unione tale che $j \in T_k$ e $Q := \bigcup_{h \in H} T_h$

Capitolo 5

Interpretazione della SVM tramite SHAP

Questo capitolo verterà sull'applicazione concreta dei metodi di stima precedentemente discussi, attraverso l'utilizzo della libreria SHAP (SHapley Additive exPlanations), ideata da Scott Lundberg [1] Un aspetto fondamentale di SHAP è rappresentato dagli Explainer, ovvero funzioni dedicate all'implementazione dei vari metodi di stima che, come input, ricevono:

■ model: una funzione che, dato un insieme di dati, restituisce le previsioni del modello che si sta cercando di spiegare. Per alcuni modelli (come per l'SVM) è necessario passare direttamente all'explainer il metodo model.predict, per evitare di riceve un messaggio di errore dovuto al fatto che l'explainer non sia in grado di gestire il modello. Questo non significa che il modello non sia spiegabile, ma che in SHAP non ci siano i passaggi per raggiungere il .predict del model dato in input

■ masker: nel paragrafo 4.2.1 si è visto come durante il calcolo dei valori di SHAP sia necessario campionare i valori delle feature assenti da una coalizione da una parte del dataset il *background data*. Tale campionamento, nella libreria, viene effettuato da degli oggetti che prendono il nome di masker. In SHAP ci sono diversi tipi di masker, in questo lavoro ne verranno trattati due:

- Independent: è il masker usato tipicamente per i dati tabulari. Quando chiamato da un explainer, effettua la sostituzione delle feature mancanti utilizzando dei campioni estratti dal background
- Partition: è un masker utilizzato principalmente nel Partition Explainer, visto il suo tenere conto della struttura gerarchica delle feature del dataset

Costituisce buona pratica utilizzare parte del training set come background data, mentre i valori di SHAP vengono calcolati sul test set. Tale distinzione permette di evitare che per certi campioni feature mancanti vengano sostituiti con i valori dell'esempio stesso. Con il crescere della dimensione del background si ottengono risultati più accurati, a discapito di un rallentamento dell'explainer. Gli autori dello studio [10], fatto per determinare quanto significativo sia l'impatto di tale effetto, consigliano di usare un background quanto più largo possibile per ridurre il problema dell'oscillazione dei valori di SHAP in caso di un numero limitato di campioni. In questa tesi considerando la piccola dimensione dataset si è preferito limitarsi ad un insieme di 100 elementi. In generale, sia " t " il numero di esempi della porzione di dataset che si vuole spiegare. Applicare un Explainer a tali dati restituisce in output un Explanation Object, ovvero un insieme di tre array:

■ values: $t \times n$ array, contenente i valori di SHAP

■ base_values: $t \times 1$ array, contenente la ripetizione della media delle previsioni del modello sui campioni del background $E[f(X)]$

■ data: $t \times n$ array, contenente i valori dei dati per cui si stanno calcolando i valori di SHAP

Dopo aver definito un explainer, per esempio:

```
Permutation_explainer = shap.explainers.Permutation(svm_model.predict, background_data)
```

ci sono due modi per creare un explanation object:

■ METODO DIRETTO: si chiama direttamente l'explainer su ciò che si vuole spiegare:

```
Explanation_object = Permutation_explainer(X_test)
```

■ METODO INDIRETTO: si utilizza il metodo .shap_values per calcolare i valori di SHAP e si crea successivamente manualmente l'explanation object utilizzando la funzione .Explanation:

```
shap_values = Permutation_explainer.shap_values(X_test)
shap.Explanation(shap_values, base_values, X_test)
```

Dove base_values è il vettore che ripete t-volte il valore:

```
svm_model.predict(background_sample).mean()
```

I due metodi producono risultati simili. In particolare, quello indiretto risulta essere più veloce rispetto alla sua controparte, probabilmente a causa di ulteriori approssimazioni che avvengono in .shap_values

Nel capitolo 3 si è utilizzato una pipeline per gestire in serie la standardizzazione del dataset con la ricerca tramite GridSearch dei parametri del modello migliore. Gli Explainer Agnostici della libreria sono in grado di interpretare qualsiasi funzione, anche una pipeline, ma in questo caso interessa spiegare la migliore SVM trovata. Quindi dal .best_estimator della GridSearch si estrae lo StandarScaler e il modello, usando rispettivamente .named_steps['scaler'] e named_steps['svc']

Come background data sono stati estratti casualmente 100 campioni da X_train (la parte di dataset del training set, dopo che è stata trasformata tramite lo scaler). In questa parte di studio si è deciso di utilizzare il Permutation Explainer, lasciando il masker di default di SHAP (ovvero l'indipendent masker). Come metodo per la creazione dell'explanation object si è ricorso al metodo indiretto, chiamando quindi il modulo .shap_values su X_test (la parte di dataset del test set, dopo che è stata trasformata tramite lo scaler). Nel campionamento e nella dichiarazione dell'explainer, per garantire la riproducibilità dei risultati si è impostato il random_seed = 246120.

Uno dei punti di forza della libreria di SHAP risiede in una vasta scelta di diversi plot che permettono di analizzare il modello in maniera approfondita. Tali plot possono essere divisi in due macrogruppi a seconda del loro impiego per spiegazioni locali di singole istanze, o per interpretazioni globali.

5.1 Interpretazione Singole Previsioni

Selezionando specifiche righe dell'Explanation Object è possibile rappresentare le spiegazioni locali di singole istanze di dati. In SHAP ci sono due tipi principali di plot per questo tipo di interpretazioni:

■ Waterfall Plot

Il primo plot prende il nome di *Waterfall Plot* visto il suo ricordare una cascata. Nell'asse-y vengono mostrate le varie feature, disposte dal basso verso l'alto in ordine crescente di contributo assoluto alla previsione. Ogni barra orizzontale rappresenta il valore di SHAP per quella specifica feature. Il grafico parte dalla previsione media $E[f(X)]$ e cresce/decresce fino a raggiungere la previsione del modello. Barre di color rosso indicano contributi positivi, mentre quelle blue sono contributi negativi.

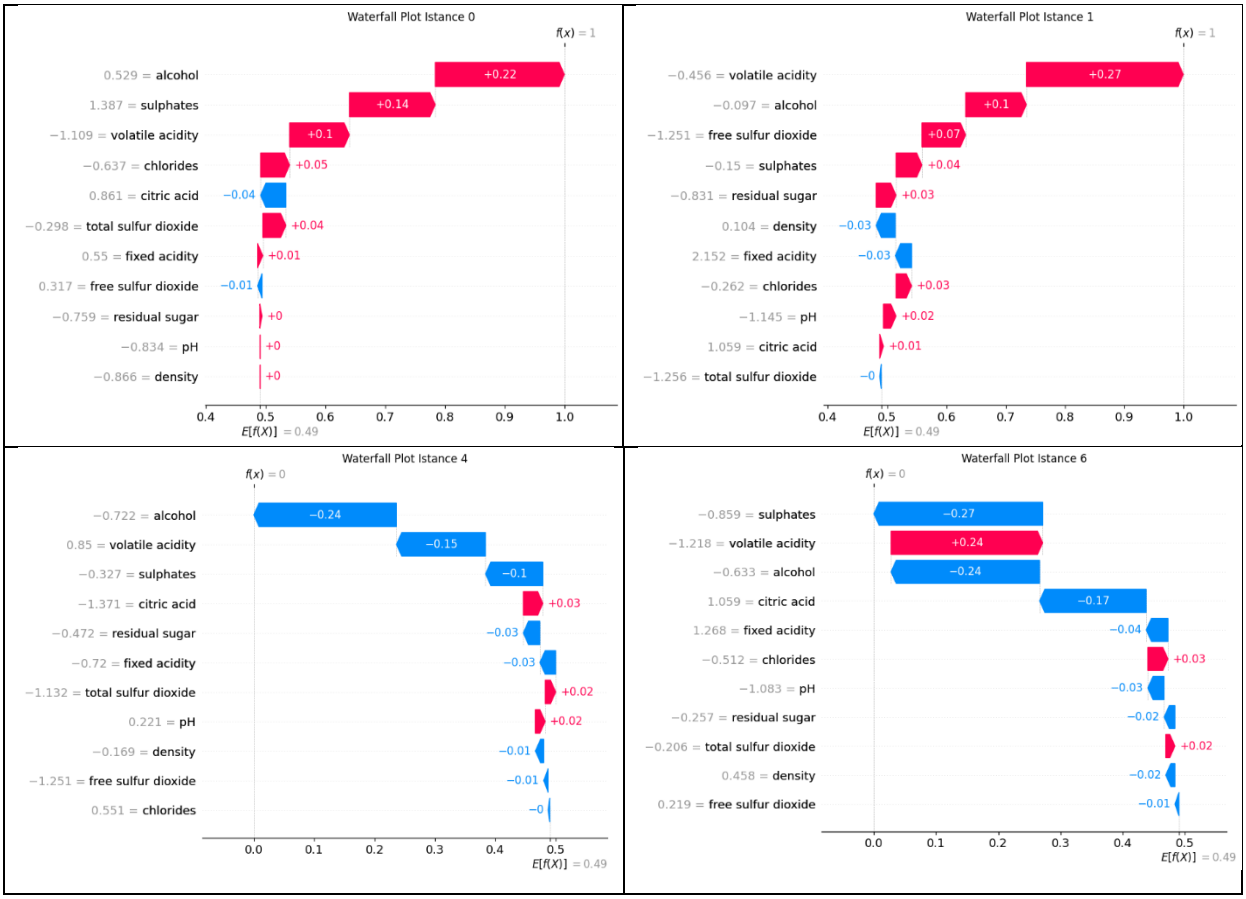


Figura 5.1 - Confronto fra i Waterfall Plot di vari campioni

Nella Figura 5.1 sono riportati i plot di alcuni campioni del `X_test`. In particolare, i primi due subplots mostrano i valori di SHAP corrispondenti alle istanze per le quali il modello ha predetto l'appartenenza alla classe 1, ovvero vini di qualità superiore alla media. Gli altri due subplots, invece, riguardano campioni di vino classificati con una qualità inferiore. Da un confronto dei grafici di diversi campioni, emerge che, sebbene il waterfall plot offra una facile comprensione, presenta il limite di non essere ottimale per fornire interpretazioni consistenti. Infatti, il plot dell'istanza 0 porterebbe a pensare che le variabili più decisive per la qualità dei vini siano il grado alcolico e i solfati, mentre nel plot dell'istanza 1, l'impatto maggiore per la previsione è dato dai livelli di acidità volatile.

■ Force Plot

Il *Force Plot* svolge un ruolo simile al Waterfall Plot. I valori di SHAP di ogni feature sono rappresentati in un'unica barra orizzontale come delle frecce convergenti verso il centro. Le feature vengono, infatti, viste come delle "forze" che contribuiscono positivamente o negativamente alla previsione, il loro punto di bilanciamento.

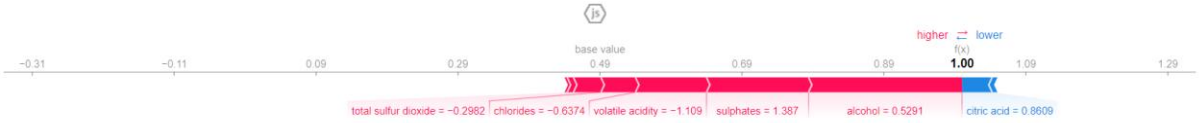


Figura 5.2 - Force Plot Istanza 0

5.2 Spiegazione Globale del Modello

L'interpretazione globale si concentra sul comportamento medio del modello, mostrando come le feature contribuiscano alla previsione e come interagiscono fra loro.

■ Summary Plot

Mentre il waterfall plot evidenzia i contributi delle varie feature per un singolo campione, il *Summary Plot* (o altrimenti noto come *Beeswarm Plot*) consente di visualizzare l'andamento complessivo dei valori di SHAP. Ogni variabile è rappresentata da una riga nel grafico, con i colori che, a differenza dei plot locali, riflettono i valori delle feature.

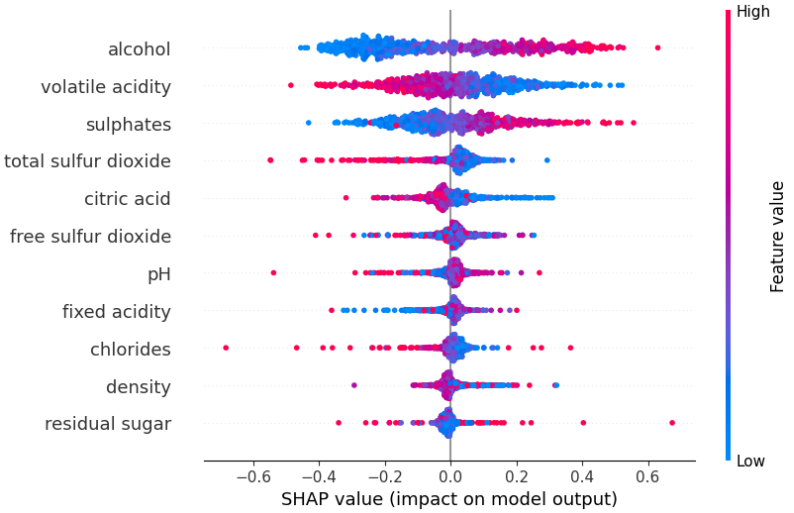


Figura 5.2 - Summary Plot

L'ordine delle feature è determinato dalla loro *importanza*, ovvero la media dei valori di SHAP:

$$I_j = \frac{1}{t}\sum_{i=1}^{t} \phi_j^{(i)} \tag{5.1}$$

con t che indica il numero di campioni della parte di dataset che si sta spiegando

Nel caso preso in considerazione, emerge chiaramente come alcol, acidi volatili e solfati siano le feature più rilevanti per la qualità di campioni di vino. Però, è fondamentale evitare di cercare osservazioni causali durante l'interpretazione dei valori di SHAP. Il fatto che l'alcol sia la variabile con un'importanza più alta non significa che questo si verifichi anche nella realtà, ma indica piuttosto che tale feature è cruciale per le previsioni del modello. Una più corretta valutazione potrebbe nascere dalla conoscenza del settore. Infatti, come riassunto nella Tabella 3.1, l'acidità di un vino è intrinsecamente legata alla qualità. In generale livelli moderatamente bassi di acidità volatile ravvivano il vino, contribuendo alla sua complessità olfattiva, mentre alti valori possono avere un impatto troppo forte, conferendo odori che tendono all'aceto. Il summary plot evidenzia come il modello possieda una giusta bias nei confronti di 'volatile acidity'. In particolare, si osserva che alti valori di tale feature (rappresentati da punti colorati in rosso) sono correlati a valori di SHAP molto negativi, indicando il significativo contributo della variabile nella decisione della SVM di associare campioni acidi a una qualità inferiore alla media. Analisi analoga può essere fatta per i solfati, che essendo conservanti permettono di allungare la vita del vino preservandone il sapore.

■ Dependence Plot

Il *Dependence Plot* (o altrimenti noto come *Scatter Plot*) può essere usato per esaminare le relazioni fra le feature, sfruttando una quantità che prende il nome di *indice di interazione di Shapley*:

$$\phi_{j,k} = \sum_{S \subseteq N \setminus \{j,k\}} \frac{|S|!\,(n - |S| - 2)!}{2(n-1)!} \delta_{jk}(S)$$

dove $j \neq k$ e $\delta_{jk}(S) = v(S \cup \{j,k\}) - v(S \cup \{j\}) - v(S \cup \{k\}) + v(S)$

Quando utilizzato per illustrare una feature, il dependence plot chiama approximate_interactions, una funzione di SHAP che calcola i vari indici di interazione, trovando quello relativo alla variabile più legata a quella che si sta cercando di rappresentare. Si prenda in considerazione, ad esempio, il dependence plot di 'alcohol' , la feature di maggiore importanza:

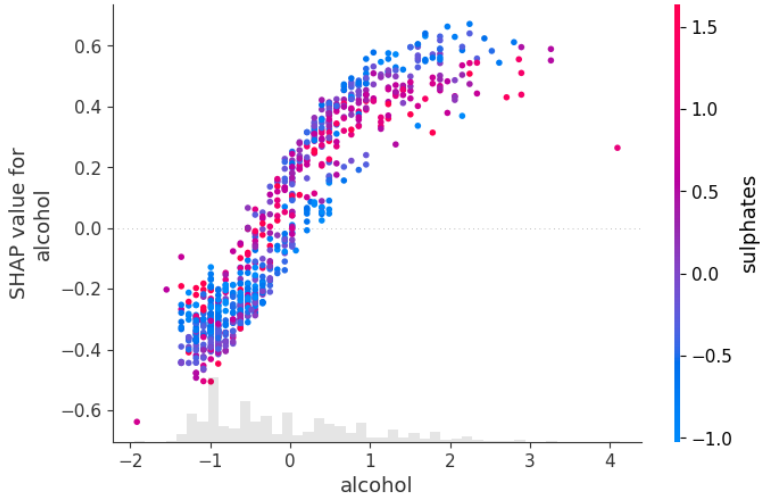


Figura 5.3 - Dependence Plot 'alcohol'

Analizzando la Figura 5.3, si può notare che per alti livelli di alcol, i campioni con minori quantità di solfati presentano, a parità di gradazione, valori di SHAP più grandi. Questa relazione si inverte invece per vini con una più bassa percentuale alcolica.

5.3 Il Problema della Correlazione

La presenza di correlazione tra le feature del dataset può causare complicazioni durante il calcolo dei valori SHAP. Simulare l'assenza di tali variabili, mediante l'estrazione di valori mancanti dal background, conduce alla creazione di campioni poco realistici. Ad esempio, si immagini che il dataset "winequality" contenga due colonne aggiuntive indicanti rispettivamente il mese e il giorno di imbottigliamento dei vini. Estrarre casualmente dati potrebbe generare istanze con date impossibili, come il 30 febbraio. Queste irregolarità oltre che rappresentare un problema concettuale causano il modello a commettere previsioni errate, visto il suo non essere allenato su tali dati. Tale processo di effettuare stime al di fuori di intervalli osservati prende il nome di *estrapolazione*. Un altro problema risiede nel fatto che, da un punto di vista della teoria dell'informazione, feature correlate condividono *informazione*. Tale informazione viene persa quando, durante il calcolo dei contributi di coalizioni, si campiona solo una parte degli insiemi di feature correlate.

Una semplice soluzione potrebbe consistere nell'eliminare le feature correlate e riallenare il modello sul nuovo dataset prima di calcolare i valori di SHAP. Questo non è però sempre possibile. Ne è un esempio il dataset wine quality, dove il numero di variabili è troppo basso per giustificare eventuali tagli. In questi casi è più conveniente adottare un approccio alternativo che si basi sull'impiego del *Partition Explainer*. Come metrica per la partizione si può usare l'*indice di correlazione di Pearson*:

$$r_{xy} = \frac{\sum_{j=1}^{n}(x_j - \bar{x})(y_j - \bar{y})}{\sqrt{\sum_{j=1}^{n}(x_j - \bar{x})^2}\sqrt{\sum_{j=1}^{n}(y_j - \bar{y})^2}}$$

dove $(x_1,\ldots, x_n)$ *e* $(y_1,\ldots, y_n)$ sono due campioni aventi come medie $\bar{x} = \frac{1}{n}\sum_{j=1}^{n} x_j$ e $\bar{y} = \frac{1}{n}\sum_{j=1}^{n} y_j$

Le feature vengono quindi divise in gruppi all'interno di una struttura ad albero, seguendo un ordine gerarchico dettato dall'indice. Successivamente, vengono passate ad un Partition Explainer che ne calcola i valori di Owen (vedi paragrafo 4.3.3). Questo procedimento non risolve completamente il problema della correlazione, visto che man mano che si scende verso il basso dell'albero si rompono gruppi di feature molto correlate fra loro. Però l'effetto dell'estrapolazione è notevolmente ridotto. Un importante accorgimento è però necessario. Il clustering basato sulla correlazione porterebbe le feature con una forte correlazione negativa ad far parte di gruppi diversi. Per evitare tale problema la gerarchia delle variabili verrà fatta sulla *correlazione assoluta*. Il risultato sarà un albero dove le feature con un'alta correlazione, a prescindere che positiva o negativa, vengono raggruppate.

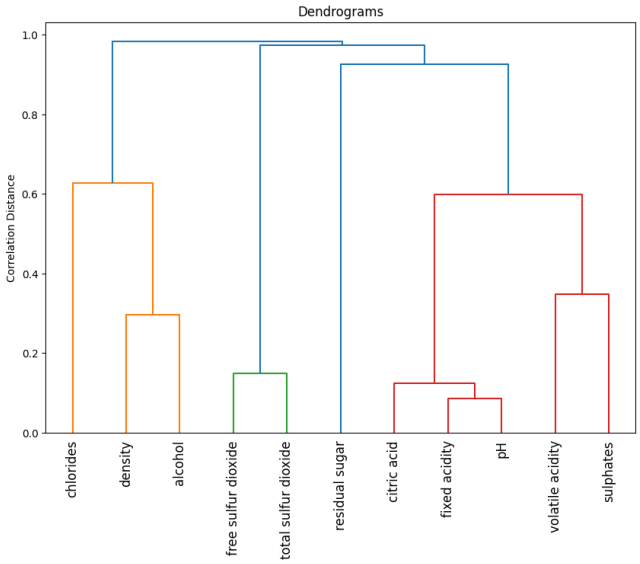


Figura 5.4 - Hierarchically Clustering delle feature

La Figura 5.4 mostra i risultati del clustering. La correlazione più forte è quella fra gli acidi fissi e il pH. Le due variabili vengono combinate fra di loro per poi essere unite prima ai livelli di acido citrico e poi al gruppo formato da acidi volatili e solfati. Man mano che si procede verso l'alto nel grafico, la correlazione parte dal suo massimo, raggiunto dalle singole feature, per poi diminuire con il crescere della distanza fra i gruppi. Ulteriori relazioni interessanti sono emerse tra i solfuri liberi e totali, così come tra la densità, i livelli di alcol e i livelli di cloruro. Questi risultati sono in sintonia con quanto ci si potrebbe aspettare da campioni di vino. La gerarchia dei clustering viene fornita in input, agendo come masker, a un Partition Explainer. Quest'ultimo la utilizza per determinare i valori SHAP, calcolati in base ai valori di Owen.

■ Importance Plot

Dopo aver calcolato i nuovi valori di SHAP è immediato chiedersi se i risultati trovati differiscono da quelli ottenuti precedentemente con il permutation explainer. Per questo confronto si può utilizzare l'*Importance Plot* (o altrimenti noto come *Bar Plot*) della libreria SHAP. Tale plot permette di rappresentare un ranking delle feature basato sulla loro importanza (Formula (5.1))

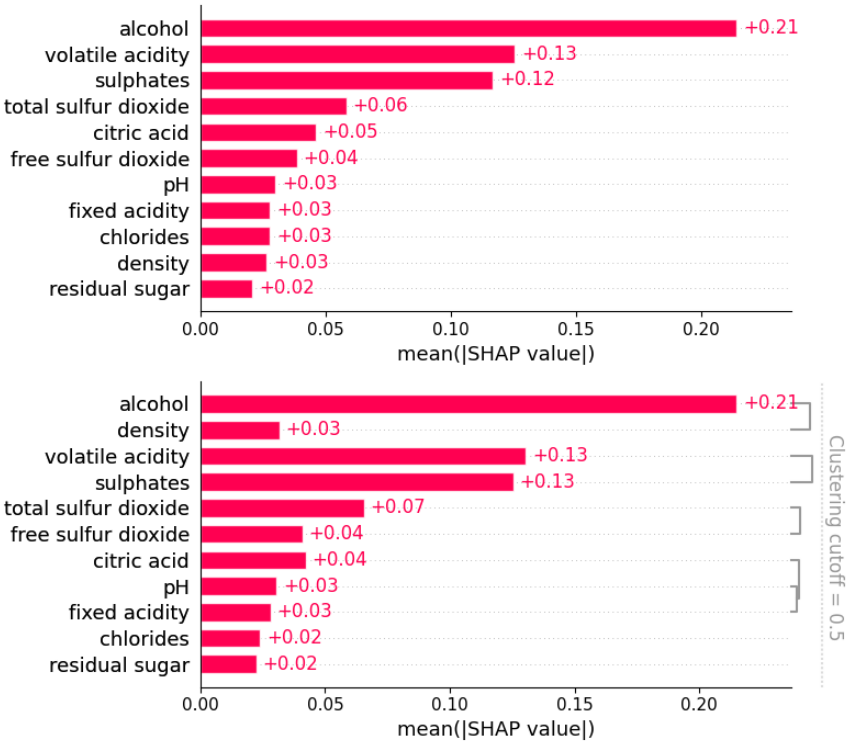


Figura 5.5 - Confronto Importance Plot dei due Explainer

Dalla Figura 5.5 si può osservare come la differenza fra le importanze dei due grafici risulti essere minima, indicando che, per il dataset wine quality, la correlazione tra le caratteristiche non ha influito significativamente sulle capacità esplicative di SHAP. Nonostante ciò, va sottolineato che il lavoro svolto ha comunque il merito di introdurre nuove interpretazioni. Guardando il secondo plot, infatti, si può vedere come le feature vengono unite in cluster. Questo significa che invece che considerarli singolarmente, i valori di SHAP di gruppi di feature possono essere sommati. Pertanto, è possibile esaminare i contributi alla previsione dei cluster con la certezza che i calcoli eseguiti non abbiano causato estrapolazione. Il parametro clustering_cutoff del bar plot permette di regolare la distanza di correlazione che le feature non devono superare per essere considerate sufficientemente vicine da appartenere ad un cluster. In generale, insiemi più ampi sono associati a una minore correlazione, anche se questo può tradursi in un raggruppamento meno significativo.

Conclusioni

In conclusione, questa tesi ha esplorato con successo l'applicazione dei Valori di Shapley e dei Valori di SHAP come strumenti chiave nella spiegazione e interpretazione dei modelli di machine learning. Attraverso l'analisi di un dataset e l'implementazione pratica di SHAP ad una SVM, si è dimostrato come questo approccio possa migliorare la comprensione delle complesse decisioni del modello. Quello che distingue SHAP da altri metodi interpretativi è la sua solida base teorica: l'assioma di efficienza garantisce, infatti, una distribuzione equa delle previsioni medie tra i vari giocatori. Questi motivi hanno conferito a SHAP una notevole popolarità negli ultimi anni. Tuttavia, è fondamentale sottolineare che, come ogni strumento, SHAP presenta alcune limitazioni. Nel paragrafo 5.3 è stato già discusso di come la correlazione possa influenzare le previsioni e, di conseguenza, fornire spiegazioni sbagliate. In aggiunta, è importante notare che il costo computazionale di questo metodo cresce in modo esponenziale con il numero di feature, per via del dover gestire 2^n coalizioni. Negli ultimi anni tale sfida è stata leggermente mitigata grazie all'introduzione di Explainer alternativi, come il Permutation, che hanno permesso di produrre interpretazioni model-agnostic in tempi più brevi rispetto alla prima iterazione di SHAP, il Kernel Explainer. Nei casi di dataset estremamente grandi potrebbe essere però necessario ricorrere ad algoritmi più specializzati, come il Tree Explainer o il Deep Explainer. Un ulteriore svantaggio del metodo risiede nel fatto che SHAP non offre una prospettiva "*what-if*". Ciò implica che l'utente, durante l'analisi delle interpretazioni, non dispone di indicazioni *controfattuali* che gli permettano di capire quali modifiche apportare o quali azioni intraprendere per migliorare le previsioni del modello. Infine, i valori di SHAP si limitano a spiegare quali sono le feature più importanti, non forniscono quindi interpretazioni *causali* applicabili nella realtà. Ad esempio, dai risultati ottenuti emerge che campioni con elevate percentuali di alcol sono in generale associati a una qualità superiore del vino, ma ciò non significa che aumentare la gradazione possa automaticamente garantire dei prodotti migliori. Per risolvere tali problemi sono nate molte estensioni come Counterfactual SHAP [11] o Causal Shapley Values [12] che si pongono l'obiettivo di ampliare SHAP, cercando di colmare le sue limitazioni. Con la continua evoluzione di tecniche e strumenti di spiegazione, sarà quindi possibile affrontare le sfide attuali e migliorare ulteriormente la comprensione e l'interpretazione dei modelli di machine learning. In conclusione, SHAP rappresenta un passo significativo verso una maggiore trasparenza e interpretabilità nella crescente complessità del panorama dell'intelligenza artificiale.

Bibliografia

[1] LUNDBERG SM, LEE S-I. (2017) *A unified approach to interpreting model predictions.* In: Guyon I, Luxburg UV, Bengio S, et al. (eds) *Advances in neural information processing systems*

[2] GILLESPIE, N., LOCKEY, S., CURTIS, C., POOL, J., & AKBARI, A. (2023). *Trust in Artificial Intelligence: A Global Study*, The University of Queensland and KPMG Australia. 10.14264/00d3c94

[3] MOLNAR, C. (2022). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable (2nd ed.)* christophm.github.io/interpretable-ml-book/

[4] SCHOLBECK, C. A., MOLNAR, C., HEUMANN, C., BISCHL, B., & CASALICCHIO, G. (2019). *Sampling, Intervention, Prediction, Aggregation: A Generalized Framework for Model-Agnostic Interpretations.* ArXiv. https://doi.org/10.1007/978-3-030-43823-4_18

[5] MOLNAR, C. (2023*). Interpreting Machine Learning Models With SHAP A Guide With Python Examples And Theory On Shapley Values*

[6] CORTEZ,PAULO, CERDEIRA,A., ALMEIDA,F., MATOS,T., AND REIS,J. (2009). *Wine Quality. UCI Machine Learning Repository.* https://doi.org/10.24432/C56S3T.

[7] DIRK P. KROESE, ZDRAVKO I. BOTEV, THOMAS TAIMRE, RADISLAV VAISMAN (2023) *Data Science and Machine Learning: Mathematical and Statistical*, CRC Press

[8] OWEN, G. (1977). *Values of games with a priori unions. In Essays in Mathematical Economics and Game Theory*, pages 76–88 R. Henn and O. Moeschelin (Editors), Springer-Verlag

[9] PUENTE, M. (2019). *The Owen and the Owen-Banzhaf Values Applied to the Study of the Madrid Assembly and the Andalusian Parliament in Legislature 2015-2019*. Proceedings of the 8th International Conference on Operations Research and Enterprise Systems.

[10] YUAN, H., LIU, M., KANG, L., MIAO, C., & WU, Y. (2022). *An empirical study of the effect of background data size on the stability of SHapley Additive exPlanations (SHAP) for deep learning models. ArXiv*. /abs/2204.11351

[11] ALBINI E, LONG J, DERVOVIC D, MAGAZZENI D (2022) *Counterfactual shapley additive explanations. In: 2022 ACM conference on fairness, accountability, and transparency. pp 1054–1070*

[12] HESKES, T., SIJBEN, E., BUCUR, I. G., & CLAASSEN, T. (2020). *Causal Shapley Values: Exploiting Causal Knowledge to Explain Individual Predictions of Complex Models*. *ArXiv*. /abs/2011.01625