UNIVERSITÀ
DEGLI STUDI
DI BERGAMO

Dipartimento di Ingegneria e Scienze Applicate

Corso di Laurea Magistrale in Ingegneria Informatica

Un modello di ottimizzazione mista intera per un problema di classificazione di dati COVID-19

Relatore:
Chiar.ma Prof.ssa Francesca Maggioni

Correlatore:
Chiar.ma Prof.ssa Maria Teresa Vespucci

Tesi di Laurea Magistrale

Riccardo Allegri
Matricola n. 1046242

Anno Accademico 2022/2023

Indice

Abstract

Negli anni 2020-2021, a causa dalla pandemia di SARS-CoV-2, il Servizio Sanitario Nazionale ha dovuto affrontare situazioni di forte stress dovute principalmente al sovraffollamento degli ospedali che ha causato uno squilibrio tra richiesta e disponibilità di assistenza.

A causa della scarsità di risorse sanitarie, i medici quotidianamente erano sottoposti a una forte pressione dovuta alla responsabilità di assegnare una priorità alla cura dei pazienti.

L'obiettivo principale di questa tesi è di proporre un algoritmo di *machine learning* con lo scopo di supportare i medici nel prevedere e calcolare il rischio di mortalità per Covid-19 al fine di indirizzarne in tempi rapidi il percorso terapeutico più appropriato, ovvero, in ottica clinica, di "stratificare il rischio di morte" [9].

Nel primo capitolo viene fatta una breve introduzione sulla malattia COVID-19 e viene fornita una panoramica iniziale sui dati a disposizione e su come questi sono stati reperiti.

Nel secondo capitolo viene introdotto l'algoritmo di *machine learning "Optimal Classification Trees"* di Dimitri Bertsimas e Jack Dunn [2], un algoritmo di classificazione che si discosta da quelli usuali in quanto utilizza tecniche di programmazione matematica in sostituzione di quelle *greedy* comunemente in uso.

Nel terzo capitolo viene introdotto il linguaggio di programmazione matematica *AMPL* e come il modello è stato implementato in tale linguaggio.

Nel quarto capitolo si parlerà dei metodi usati per la validazione, facendo particolare attenzione a quello usato per questa tesi.

Nel quinto capitolo verranno esposti i risultati computazionali.

Infine nel sesto capitolo verranno introdotte le tecniche per la robustificazione del modello.

Capitolo 1

Introduzione

La COVID-19, conosciuta anche come sindrome respiratoria acuta grave da SARS-CoV-2 o malattia da coronavirus, è una malattia infettiva respiratoria causata dal virus denominato SARS-CoV-2 appartenente alla famiglia dei coronavirus [11].

I coronavirus sono virus che circolano naturalmente tra gli animali e alcuni di essi possono infettare anche l'uomo.

I pipistrelli sono considerati gli ospiti naturali di questi virus, ma anche molte altre specie di animali sono considerate fonti. Ad esempio, il Coronavirus della sindrome respiratoria del Medio Orientale (MERS-CoV) viene trasmesso all'uomo dai cammelli e la sindrome respiratoria acuta grave Coronavirus-1 (SARS-CoV-1) viene trasmessa all'uomo dallo zibetto.

Il canale di diffusione principale di SARS-CoV2 è quello umano, dove una persona infetta starnutisce, tossisce, parla o respira e si trova in prossimità di altre persone. Infatti, le goccioline possono essere inalate o possono poggiarsi su superfici, con cui altri vengono a contatto e vengono quindi incorporate attraverso il contatto di naso, bocca o occhi. Il virus può sopravvivere su superfici per poche ore (rame, cartone) fino a un certo numero di giorni (plastica e acciaio inossidabile), tuttavia la quantità di virus vitale diminuisce nel tempo e potrebbe non essere sempre presente in quantità sufficiente da causare l'infezione.

I sintomi di COVID-19 variano sulla base della gravità della malattia, dall'assenza di sintomi (essere asintomatici) a presentare febbre, tosse, mal di gola, debolezza, affaticamento e dolore muscolare. I casi più gravi possono presentare polmonite, sindrome

da distress respiratorio acuto e altre complicazioni, tutte potenzialmente mortali. Perdita improvvisa dell'olfatto (anosmia) o diminuzione dell'olfatto (iposmia), perdita del gusto (ageusia) o alterazione del gusto (disgeusia) sono stati riconosciuti come sintomi di COVID-19. Altri sintomi meno specifici possono includere cefalea, brividi, mialgia, astenia, vomito e/o diarrea [11].

Al 6 aprile 2020 (inizio pandemia), secondo i dati dell'Organizzazione Mondiale della Sanità (OMS), nel mondo erano documentati 554.550 casi di COVID-19 con 47.687 decessi, mentre ad oggi (06/03/2023) il numero di contagiati totale è salito a 758.390.564 mentre quello dei decessi a 6.859.093 [3].

1.1 Dati iniziali

I dati oggetto di studio provengono dal Pronto Soccorso dell'Ospedale di Valcamonica (Esine, Brescia, Lombardia, Italia) e riguardano un gruppo di 150 pazienti ricoverati per Covid-19 tra il 5 marzo e il 1° aprile 2020.

Questi dati sono stati divisi in due diverse categorie, la prima comprende 78 pazienti sopravvissuti e dimessi, mentre il secondo gruppo comprende 72 pazienti deceduti nel corso della loro degenza ospedaliera. A tutti è stato diagnosticato il COVID-19 secondo gli standard attuali, cioè mostrando risultati suggestivi alla tomografia computerizzata del torace (TC; con il classico *pattern* "a vetro smerigliato") e risultati positivi di reazione a catena della polimerasi inversa (RT-PCR) per SARS-CoV-2.

Per ogni paziente, inoltre, al momento del ricovero ospedaliero, sono stati registrati tutti i dati clinici e di laboratorio secondo le procedure di controllo interno della qualità (IQC) e attraverso la partecipazione allo schema di valutazione esterna della qualità (EQAS) della regione Lombardia, Italia [6].

Gli attributi considerati sono parametri vitali, comorbidità e alcuni parametri laboratoristici preventivamente selezionati in quanto ritenuti maggiormente significativi. Nello specifico sono state considerate le seguenti variabili:

- Test COVID-19 (esito positivo, negativo o dubbio).
- Sesso (Maschio/Femmina).
- Età (range: 32-95).
- Presenza di Malattie Croniche (Si/No).
- Presenza di Neoplasia (Si/No).
- Diabete (Sì/No).
- Presenza di Malattie Cardiovascolari (Sì/No).
- Immunodeficienza (Sì/No).
- Presenza di Malattie Respiratorie (Sì/No).

- Presenza di Malattie Renali (Sì/No).
- Presenza delle malattie metaboliche (Sì/No).
- Indice di massa corporea BMI nell'intervallo 30-40 (Sì/No).
- Indice di massa corporea BMI>40 (Sì/No).
- Globuli bianchi WBC (range: 2,18-25,53).
- Conteggio piastrine PLT (range: 64-521).
- Neutrofili (range: 1,31-21,43).
- Linfociti (range: 0,27-2,77).
- D-dimero (range: 270-20000).
- Aspartato aminotransferasi AST (range: 9-464).
- Lattato deidrogenasi LDH (range: 117-1161).
- Creatina chinasi CK (range: 7-4038).
- Proteina C-reattiva PCR (range: 8,3-372,6).
- Troponina I cardiaca ad alta sensibilità cTnI (range: 5-5124).
- Ferritina (range: 127-8413).
- WBC/linfociti %(range: 1,78-46,95).
- Emogas P/F (range: 36-687,14).

Poiché un numero troppo elevato di attributi in input può causare overfitting e conseguenti scarse prestazioni degli algoritmi di apprendimento automatico, è stato prima eseguito un test chi-quadrato [6] che consente di ridurre il numero di variabili in input identificando se ciascuna variabile di predizione è indipendente dalla variabile di risposta (Sopravvissuto/Deceduto).

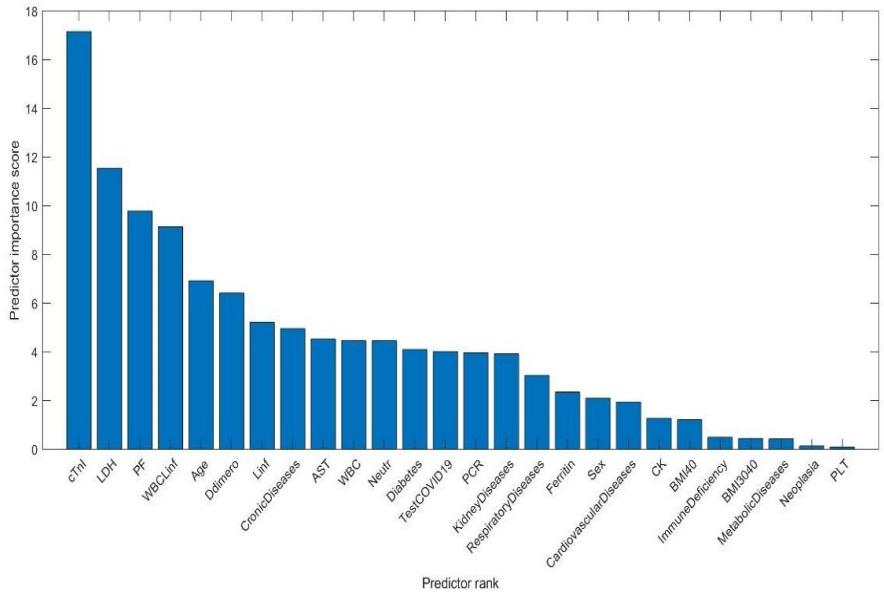

Figura 1.1: Risultati test chi-quadro.
Fonte: Un algoritmo di apprendimento automatico per l'ottimizzazione delle cure di pazienti covid-19 di Maggioni F., Gheza F., Manelli F., Bonetti G. Atti dell'Ateneo di Scienze Lettere e Arti di Bergamo.

Come mostrato in figura 1.1 i risultati del test hanno identificato che i 15 attributi di maggior rilevanza sono:

- cTnI.
- LDH.
- P/F.
- WBC/Linf.
- Età.
- Ddimero.
- presenza di malattie croniche.
- AST.
- Neutr.

- Diabete.
- Test COVID-19.
- PCR.
- Presenza di Malattie Renali.
- CK.
- Presenza di Malattie Respiratorie.

I parametri appena esposti rappresentano informazioni sensibili per il paziente, di conseguenza sono state raccolte previa approvazione del Comitato Etico di Brescia (certificato n. NP 4036) ed in conformità con la dichiarazione di Helsinki.

Capitolo 2

Albero di decisione ottimizzato

In questo capitolo verrà prima introdotto il concetto di classificazione portando in particolare l'interesse sugli alberi di decisione, successivamente verrà mostrato come si costruisce un albero di decisione ottimizzato o "*Optimal Classification Tree* (OCT)" nel caso univariato, seguirà poi la naturale estensione al caso multivariato.

2.1 Introduzione ai problemi di classificazione usando gli alberi di decisione

Prima di parlare nello specifico dei problemi di classificazione ed in particolare degli alberi di decisione viene fornito il contesto in cui sono inserite queste tipologie di problemi.

La classificazione è un insieme di tecniche che fan parte dell'apprendimento supervisionato, il quale è una branca del *machine learning* che a sua volta fa parte dell'intelligenza artificiale.

Non esiste una definizione univoca che permette di descrivere cos'è l'intelligenza artificiale; negli anni ne sono state proposte diverse, tutte accettate in ambito accademico. Il nesso che sussiste in tutte queste definizioni è l'attribuzione ai calcolatori di capacità paragonabili a quelle del cervello umano, ovvero l'intelligenza artificiale viene definita come una disciplina che studia se e in che modo si possano realizzare sistemi informatici intelligenti in grado di simulare la capacità e il comportamento del pensiero umano.

Branca dell'intelligenza artificiale è l'apprendimento automatico (*machine learning*) che viene cosi definito da Tom M. Mitchell (informatico e professore della Founders University presso la Carnegie Mellon University, uno dei massimi esponenti di quest'ambito):

"Si dice che un programma apprende dall'esperienza E con riferimento a alcune classi di compiti T e con misurazione della performance P, se le sue performance nel compito T, come misurato da P, migliorano con l'esperienza E."

Questa definizione è rilevante poiché fornisce una definizione operativa dell'apprendimento automatico ovvero che, un programma apprende, se c'è un miglioramento delle prestazioni dopo che ha svolto un compito.

L'apprendimento automatico si può suddividere concettualmente in:

1. Apprendimento supervisionato.
2. Apprendimento non supervisionato.
3. Apprendimento per rinforzo.

Nell'apprendimento supervisionato si parte da un insieme di elementi (vettori) dei quali si conosce la classe di appartenenza e si costruisce un modello che sarà poi in grado di riflettere quali sono le caratteristiche dei dati, ovvero delle relazioni ingresso-uscita che permettano di assegnare un elemento ad una delle classi in questione.

L'apprendimento non supervisionato è una tecnica che consiste nel fornire al sistema informatico una serie di input (esperienza del sistema) che riclassificherà ed organizzerà sulla base di caratteristiche comuni per cercare di effettuare ragionamenti e previsioni sugli input successivi. Il compito svolto è quindi simile a quello dell'apprendimento supervisionato, con la differenza che vengono forniti al modello solo esempi non annotati, ovvero dei dati di input dei quali non si conosce la classe di appartenenza.

L'apprendimento per rinforzo (o *reinforcement learning*) è una tecnica di apprendimento automatico che punta a realizzare agenti autonomi in grado di scegliere azioni da compiere per il conseguimento di determinati obiettivi tramite interazione con l'ambiente in cui sono immersi.

A differenza delle altre due tipologie di apprendimento, questo paradigma si occupa di problemi di decisioni sequenziali, in cui l'azione da compiere dipende dallo stato attuale del sistema e ne determina quello futuro.

La qualità di un'azione è definita in termini di un valore numerico di "ricompensa", che ha lo scopo di incoraggiare comportamenti corretti dell'agente. Questo tipo di apprendimento è solitamente modellizzato tramite i processi decisionali di Markov e può essere effettuato con diverse tipologie di algoritmi, classificabili in base all'utilizzo di un modello che descriva l'ambiente, alle modalità di raccolta dell'esperienza (in prima persona o da parte di terzi), al tipo di rappresentazione degli stati del sistema e delle azioni da compiere (discreti o continui).

In tutti e tre i tipi di apprendimento i dati iniziali che si hanno a disposizione vengono suddivisi in due insiemi. L'insieme degli elementi usati per costruire il modello è solitamente chiamato *training set*, mentre l'insieme degli elementi usati per testare i risultati è chiamato *test set*.

Il primo insieme è utile per addestrare il modello, ovvero trovare dei parametri che permettano di classificare i dati di input in modo ottimale mentre il secondo serve per valutare la capacità di generalizzazione del modello, ovvero la capacità di predire corret-

tamente dati nuovi.

Per questo scopo si calcola l'accuratezza del modello che fornisce la percentuale di dati che vengono classificati in modo corretto:

$$accuratezza = \frac{TP+TN}{TP+TN+FP+FN} \cdot 100 \tag{2.1}$$

dove *TP* rappresenta il numero di veri positivi (*true positive*), *TN* i veri negativi (*true negative*), *FP* i falsi positivi (*false positive*) e *FN* i falsi negativi (*false negative*).

Possibili applicazioni sono la predizione della qualità dei fornitori o dei prodotti, la predizione della propensione all'acquisto dei clienti o come in questo lavoro di tesi predire se un paziente affetto da COVID-19 e con determinati parametri salutistici riesca a sopravvivere o meno.

Esistono diversi tipi di classificatori basati su diverse tecniche tra cui modelli di ottimizzazione a vettori di supporto, reti Bayesiane, alberi di decisione e reti neurali. Tali classificatori vengono valutati sulla base delle seguenti caratteristiche:

1. Accuratezza: percentuale elementi classificati correttamente.
2. Velocità: tempo di costruzione del modello e tempo di classificazione.
3. Scalabilità: capacità del classificatore di costruire un modello con un crescente numero di variabili.
4. Robustezza: capacità di classificazione in caso di elementi mancanti.
5. Interpretabilità: facilità nell'interpretare i risultati del modello.

Per questo lavoro la tecnica che è stata approfondita è quella degli alberi di decisione, in quanto è quella che fornisce una migliore interpretabilità dei risultati, requisito fondamentale in ambito medico.

Il tipico approccio che tenta di risolvere il problema di trovare l'albero di decisione ottimale è quello basato sulla tipologia *Classification and Regression Trees* (CART).

In particolare i dati in ingresso vengono forniti sotto forma di osservazioni $(\mathbf{x}_i, y_i),\ i = 1,..,N$, con $\mathbf{x}_i \in \mathbb{R}^p$ vettore di p caratteristiche o *features*, mentre $y_i \in \{0,..,K\}$ rappre-

senta l'etichetta assegnata a ciascun punto $\mathbf{x}_i$, altrimenti detto, la classe di appartenenza del vettore.

Senza perdita di generalità si può assumere che i valori di ciascuna dimensione dei dati di addestramento siano normalizzati nell'intervallo [0,1], ovvero $\mathbf{x}_i \in [0,1]^p$, questo per permettere una più facile scrittura del modello.

Gli alberi di decisione tradizionali creano ricorsivamente delle partizioni di $\mathbb{R}^p$ ($[0,1]^p$ in questo caso) ovvero dei sottoinsiemi disgiunti tali per cui la loro unione dia come risultato l'insieme di partenza e dove a ciascuno dei quali è assegnata una sola classe di appartenenza.

Per far ciò viene definita la struttura dell'albero come caratterizzata da *Branch Node*, ovvero nodi nei quali è possibile applicare una ramificazione o *split* e *Leaf Node* cioè nodi foglia, in cui non è possibile applicare uno *split* e ai quali viene assegnata una classe $k \in \{0,..,K\}$.

Prendendo in considerazione i *Branch Node*, essi vengono descritti dalle variabili $\mathbf{a}$ e b, dove $\mathbf{a} \in \{0,1\}^p$ è un vettore che serve a selezionare l'attributo da cui dipenderà lo *split* e $b \in \mathbb{R}$ è uno scalare che definisce la soglia secondo la quale viene valutata una determinata caratteristica e di conseguenza la divisione che prenderà l'i-esimo punto, altrimenti detto, considerando l'i-esima osservazione $\mathbf{x}_i$ verrà valutata l'espressione $\mathbf{a}^\top \mathbf{x}_i < b$ e nel caso risultasse vera allora il vettore $\mathbf{x}_i$ seguirà il ramo sinistro di quel *Branch Node* altrimenti il destro.

Costruendo l'albero in questo modo si può facilmente vedere come ogni osservazione possa appartenere ad un solo nodo foglia, in quanto ad un nodo foglia può essere assegnata una sola classe, di conseguenza si può giungere alla conclusione che all'osservazione $\mathbf{x}_i$ verrà assegnata una singola classe, formando così una partizione dell'insieme iniziale.

Caratteristica distintiva dei metodi CART è che ad ogni fase del processo di partizionamento la divisione viene trovata in modo tale da massimizzare un determinato criterio; questo criterio solitamente consiste nel massimizzare il decremento di una certa funzione di impurità per esempio l'entropia, in modo tale che, partendo dalla radice, le divisioni che si vengono a creare passo dopo passo contengano sempre meno informazione.

Questo vuol dire che le ultime divisioni saranno più deboli delle prime nel senso che

le probabilità $P(j|h)$ (cioè la probabilità che $y_i = j, j \in \{0..K\}$, insieme delle possibili classi, dato che $\mathbf{x}_i \in h, h \in \{T\}$, insieme dei *Branch Node*) non saranno più equiprobabili bensì saranno maggiori su un sottoinsieme di etichette o classi $k \in \{0,..,K\}$.

L'algoritmo procede con il partizionamento fino a quando uno dei due seguenti criteri d'arresto viene soddisfatto:

1. Non è possibile creare una divisione che contenga almeno un numero minimo di nodi N_{min} per ogni nodo figlio.

2. I punti $\mathbf{x}_i$ di un determinato nodo appartengono tutti alla medesima classe.

Una volta completato il processo di suddivisione a ciascun nodo foglia viene assegnata un'etichetta $k \in \{0,..,K\}$. Questo assegnamento viene fatto in modo tale da rispecchiare la classe più comune per i vettori $\mathbf{x}_i$ appartenenti a quella determinata regione, ovvero si sceglie k tale che:

$$k = \arg\max_{j \in 0,..,K} \sum_{i=1,..,N} \text{classe}(\mathbf{x}_i) = j.$$

Le operazioni presentate fin'ora rappresentano il primo passaggio ovvero la costruzione dell'albero.

Il secondo passaggio consiste nella potatura. Questa seconda fase procede dai nodi foglia verso la radice e ha lo scopo di eliminare alcuni nodi riducendo così la complessità dell'albero, in modo tale da farlo generalizzare meglio, ovvero renderlo efficace nella classificazione di dati nuovi. Per far ciò viene introdotto un parametro di complessità α il quale se elevato comporta un'eliminazione maggiore di nodi con la conseguenza di ottenere un albero di dimensioni ridotte.

Questo processo di crescita e potatura in due fasi distinte è necessario quando viene usato un approccio *greedy* di tipo *top down* come in CART in quanto tener da subito in considerazione la penalità sulla complessità dell'albero potrebbe impedire all'algoritmo di selezionare una divisione debole dietro alla quale, nel successivo passaggio, potrebbe nascondersi una divisione forte.

In conclusione si può affermare che il problema che tenta di risolvere la tecnica CART può essere formulato come un problema di ottimizzazione in cui la funzione obiettivo

è caratterizzata dal *trade off* tra accuratezza e complessità e i vincoli controllano che le foglie attive contengano il numero minimo di osservazioni richiesto. Più formalmente il modello di ottimizzazione matematica per CART può essere espresso come:

$$\begin{aligned} & \min R_{xy}(T) + \alpha|T| \\ s.t. \quad & N_x(l) \geq N_{min} \qquad \forall l \in F(T) \end{aligned} \tag{2.2}$$

dove:

- $F(T)$ rappresenta l'insieme delle folgie di T;
- $R_{xy}(T)$ rappresenta l'errore di classificazione sui dati di addestramento;
- $|T|$ è il numero di nodi dell'albero T;
- $N_x(l)$ è il numero di osservazioni $\mathbf{x}_i$ appartenenti al nodo foglia l;
- N_{min} è il numero minimo di osservazioni che ogni foglia dell'albero deve contenere;
- α è il parametro usato per controllare la complessità dell'albero.

Nei successivi paragrafi verrà riformulato lo stesso problema usando le tecniche di programmazione lineare mista intera o *Mixed Integer Optimization* (MIO) le quali permetto di ottenere un'accuratezza migliore o uguale a quelle delle tecniche CART, come dimostrato da *Bertsimas e Dunn* [2].

2.2 Albero di decisione ottimizzato (caso univariato)

In questo paragrafo viene mostrato come ottenere un albero decisionale ottimo usando le tecniche di ottimizzazione mista intera (MIO); infatti ad ogni fase della creazione dell'albero si è chiamati a prendere una serie di decisioni di tipo discreto che possono essere sintetizzate come segue:

- Ad ogni nuovo nodo scegliere se ramificare o fermarsi.
- Se si decide di interrompere la ramificazione di un nodo scegliere quale etichetta assegnargli.
- Se si decide di procedere con la ramificazione scegliere su quale variabile effettuarla.
- Quando vengono classificati i punti secondo l'albero in costruzione, bisogna scegliere a quale foglia sarà assegnato un punto, in modo tale che la struttura dell'albero sia rispettata.

Formulare questo problema usando la programmazione lineare mista intera permette di modellare tutte queste decisioni discrete in un unico problema, diversamente da quanto accade nei metodi ricorsivi che considerano tali eventi decisionali isolatamente.

Modellare il processo di costruzione tramite un MIO, permette quindi di considerare l'intero impatto delle decisioni prese sin dalla cima dell'albero, contrariamente ai metodi CART dove vengono prese una serie di decisioni ottimali localmente, l'una indipendentemente dall'altra non garantendo cosi l'ottimalità globale del risultato finale.

Viene mostrato ora il problema della creazione dell'albero decisionale ottimo usando le tecniche MIO.

Fissata la profondità massima D, il massimo numero di nodi dell'albero sarà $T = 2^{(D+1)} - 1$, indicizzati tramite l'indice $t = 1, \ldots, T$. Nella figura 2.1 per esempio, si è scelto una profondità massima $D = 2$ ottenendo cosi 7 nodi.

Come nelle tecniche CART questi nodi vengono divisi in due insiemi:

- *Branch Nodes* (T_B): sono i nodi indicizzati con $t \in T_B = \{1, .., \lfloor T/2 \rfloor\}$. In questi nodi sarà valutata la possibilità di fare uno *split* e in caso positivo verrà valutata

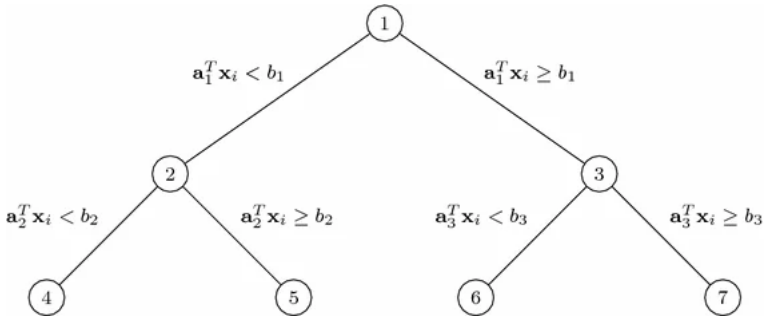

Figura 2.1: Albero decisionale di altezza 2. Fonte: Bertsimas e Dunn 2017 [2].

l'espressione $\mathbf{a}^\top \mathbf{x}_i < b$ che nel caso risulti vera porterà il punto $\mathbf{x}_i$ in questione a scegliere il ramo sinistro, nel percorso che collega la radice con una delle foglie, altrimenti il destro.

- *Leaf Nodes* (T_L): sono i nodi indicizzati con $t \in T_L = \{\lfloor T/2 \rfloor + 1, \ldots, T\}$, i quali hanno il compito di fornire una previsione o classe per ogni punto $\mathbf{x}_i$ associato a tale nodo.

Verrà usata la notazione $p(t)$ per riferirsi al nodo padre del nodo t e $A(t)$ per riferirsi all'insieme degli antenati di t. Inoltre $A_L(t)$, sottoinsieme di $A(t)$, rappresenta l'insieme degli antenati del nodo t il cui ramo sinistro è stato preso all'interno del percorso che collega la radice a t. Similmente $A_R(t)$, sottoinsieme di $A(t)$, rappresenta l'insieme degli antenati del nodo t il cui ramo destro è stato preso all'interno del percorso che collega la radice a t, ottenendo $A(t) = A_L(t) \cup A_R(t)$.

Per esempio considerando la figura 1, se il nodo in questione è $t = 5$ sì ha che $p(t) = 2$, $A_L(t) = \{1\}$, $A_R(t) = \{2\}$ e $A(t) = \{1,2\}$.

Si tiene traccia della divisione applicata al nodo $t \in T_B$ con le variabili $\mathbf{a}_t \in \mathbb{R}^p$ e $b_t \in \mathbb{R}$. Considerando il modello univariato, l'operazione di *split* può coinvolgere una sola caratteristica, ciò viene ottenuto imponendo che $\mathbf{a}_t$ siano vettori di variabili binarie che sommano a 1.

Siccome non è obbligatorio splittare ad ogni *Branch Node* viene introdotta una variabile binaria d_t la quale vale 1 se il nodo t viene splittato, 0 altrimenti. In caso si scelga l'opzione di non splittare bisogna inoltre settare $\mathbf{a}_t = 0$ e $b_t = 0$, in modo tale che l'espressione $\mathbf{a}^\top \mathbf{x}_i < b$ risulti falsa $(0 < 0)$ forzando quindi tutti i vettori $\mathbf{x}_i$ a seguire il ramo destro del nodo t. Ciò permette di arrestare anticipatamente la crescita dell'albero

in anticipo, senza introdurre nuove variabili, risparmiando di conseguenza potenza di calcolo.

Quanto detto fino ad ora viene esplicitato con i seguenti vincoli:

$$\sum_{j=1}^{p} a_{jt} = d_t, \quad \forall t \in T_B, \tag{2.3}$$

$$0 \leq b_t \leq d_t, \quad \forall t \in T_B, \tag{2.4}$$

$$a_{jt}, d_t \in \{0,1\}, \quad j = 1,..,P, \quad \forall t \in T_B. \tag{2.5}$$

Il prossimo vincolo viene introdotto per forzare la struttura gerarchica dell'albero, cioè si richiede al modello che un nodo possa splittare solo se lo fa il padre. Il motivo di questa condizione è molto semplice, infatti non ha senso splittare in un nodo dove il padre non ha splittato in quanto nel nodo figlio si presenterebbe la stessa situazione incontratasi nel nodo padre dove appunto l'algoritmo ha deciso di non splittare. Quanto detto viene espresso da:

$$d_t \leq d_{p(t)}, \quad \forall t \in T_B \backslash \{1\}. \tag{2.6}$$

Una volta definite le variabili usate per tener traccia della struttura dell'albero vengono introdotte quelle necessarie a tener traccia del numero di vettori $\mathbf{x}_i$ caduti in un determinato nodo e quelle che memorizzano lo stato di una foglia (attiva/disattiva).

Le variabili in questione sono:

- z_{it}, variabile binaria la quale vale 1 se il vettore $\mathbf{x}_i$ è nel nodo t, 0 altrimenti.
- l_t, variabile binaria la quale vale 1 se nella foglia t è caduta almeno un'osservazione, 0 altrimenti.

Queste variabili danno vita ai vincoli:

$$z_{it} \leq l_t, \quad \forall t \in T_L, \tag{2.7}$$

$$\sum_{i=1}^{n} z_{it} \geq N_{min} l_t, \quad \forall t \in T_L, \tag{2.8}$$

$$\sum_{t \in T_L} z_{it} = 1, \quad i = 1,..,N, \tag{2.9}$$

dove il vincolo (2.7) esplicita il fatto che se una foglia non è attiva ($l_t = 0$), allora nessuna osservazione vi ci può cadere dentro (z_{it} è obbligato ad assumere il valore 0 $\forall t \in T_L$), il vincolo (2.8) serve a forzare il minimo numero di $\mathbf{x}_i$ che cadono in una foglia qualora essa risulti attiva mentre il (2.9) specifica il fatto che un vettore $\mathbf{x}_i$ apparterrà solo ad una foglia e quindi ad una sola partizione dell'insieme di partenza.

E' ora necessario definire i vincoli che creano gli *split*:

$$\mathbf{a}_m^\top \mathbf{x}_i < b_m + M_1(1 - z_{it}), \quad i = 1,..,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t), \tag{2.10}$$

$$\mathbf{a}_m^\top \mathbf{x}_i \geq b_m - M_2(1 - z_{it}), \quad i = 1,..,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_R(t). \tag{2.11}$$

Sfortunatamente il vincolo (2.10) usa una disuguaglianza stretta, la quale non è supportata dai risolutori MIO, è quindi necessario introdurre una costante ε e riscrivere il vincolo nel seguente modo:

$$\mathbf{a}_m^\top \mathbf{x}_i + \varepsilon < b_m + M_1(1 - z_{it}), \quad i = 1,..,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t). \tag{2.12}$$

Tuttavia, se ε assume un valore troppo piccolo, potrebbe causare instabilità numeriche nel risolutore quindi si deve cercare un ε che sia il più grande possibile senza influenzare la fattibilità di ogni soluzione valida al problema; questo viene effettuato specificando una diversa ε_j per ogni caratteristica j. Il più grande valore valido è la più piccola distanza non nulla tra i valori della j-esima caratteristica. Per calcolare ciò si ordinano prima tali valori e poi si utilizza la formula:

$$\varepsilon_j = \min\{x_j^{i+1} - x_j^i \mid x_j^{i+1} \neq x_j^i, \quad i = 1,..,N-1\}, \tag{2.13}$$

dove x_j^i è il i-esimo valore più grande nella j-caratteristica.

Raccogliendo i valori di ε in un vettore, il vincolo di cui sopra diventa:

$$\mathbf{a}_m^\top (\mathbf{x}_i + \varepsilon) + < b_m + M_1(1 - z_{it}), \quad i = 1,..,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t).$$

Il passo successivo è quello di determinare il valore delle costanti M_1 e M_2. Siccome $\mathbf{a}_m^\top \mathbf{x}_i \in [0,1]$ e $b_m \in [0,1]$ il valore più grande che può assumere l'espressione $\mathbf{a}_m^\top (\mathbf{x}_i + \varepsilon) - b_m$ è $1 + \varepsilon_{max}$ dove: $\varepsilon_{max} = \max_j \varepsilon_j$, da cui $M_1 = 1 + \varepsilon_{max}$.

Un' operazione analoga viene fatta considerando che il valore più grande di $b_m - \mathbf{a}_m^\top \mathbf{x}_i$ risulta essere 1, da cui $M_2 = 1$.

Si possono così riassumere i vincoli finali che impongono le divisioni nell'albero:

$$\mathbf{a}_m^\top(\mathbf{x}_i + \varepsilon) + < b_m + (1 + \varepsilon_{max})(1 - z_{it}), \;\; i = 1,..,N, \;\; \forall t \in \{T_L\}, \;\; \forall m \in A_L(t), \quad (2.14)$$

$$\mathbf{a}_m^\top \mathbf{x}_i \geq b_m - M_2(1 - z_{it}), \quad i = 1,..,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_R(t). \quad (2.15)$$

Il passo successivo è di introdurre la variabile che rappresenterà l'errore di classificazione, per far ciò bisogna prima introdurre la matrice Y così definita:

$$Y := \begin{cases} +1 \quad se \quad y_i = k \\ -1 \quad altrimenti \end{cases} \qquad k = 1,..,K, \quad i = 1,..,N, \quad (2.16)$$

dove l'elemento alla riga i e colonna k vale *1* se l'osservazione i-esima appartiene alla classe k, *-1* altrimenti. La successiva variabile che viene introdotta è N_{kt} che rappresenta il numero di punti $\mathbf{x}_i$ con etichetta k nel nodo t, più formalmente:

$$N_{kt} = \frac{1}{2} \sum_{i=1}^{n} (1 + Y_{ik}) z_{it}, \quad k = 1, \ldots, K, \quad t \in T_L, \quad (2.17)$$

mentre la variabile N_t rappresenta il numero totale di punti $\mathbf{x}_i$ nel nodo t:

$$N_t = \sum_{i=1}^{n} z_{it}, \quad \forall t \in T_L. \quad (2.18)$$

E' ora necessario introdurre la variabile $c_t \in \{1,..,K\}$la quale permette di assegnare un'etichetta ad ogni foglia. Il modo più naturale per definirla è far sì che un nodo foglia acquisisca l'etichetta della classe più comune delle osservazioni che vi contiene, ovvero:

$$c_t = \underset{k=1,..,K}{\arg\max} \{N_{kt}\}. \quad (2.19)$$

Infine la variabile binaria c_{kt} (la quale vale 1 se $c_t = k$) viene introdotta per far sì che ad ogni nodo foglia che contenga almeno un punto venga assegnata una singola classe:

$$\sum_{k=1}^{K} c_{kt} = l_t, \quad \forall t \in T_L. \quad (2.20)$$

Giunti a questo punto si può quindi introdurre la variabile che rappresenta il costo di misclassificazione come la differenza tra il numero totale di punti $\mathbf{x}_i$ e il numero di punti $\mathbf{x}_i$ aventi l'etichetta più comune:

$$L_t = N_t - \max\{N_{kt}\}_{k=1,..,K} = \min\{N_t - N_{kt}\}_{k=1,...,K}. \tag{2.21}$$

Il vincolo (2.21) essendo non lineare, viene linearizzato nel seguente modo:

$$L_t \geq N_t - N_{kt} - M(1 - c_{kt}), \quad k = 1, \ldots, K, \quad \forall t \in T_L, \tag{2.22}$$

$$L_t \leq N_t - N_{kt} + Mc_{kt}, \quad k = 1, \ldots, K, \quad \forall t \in T_L, \tag{2.23}$$

$$L_t \geq 0 \quad \forall t \in T_L, \tag{2.24}$$

dove M è una costante sufficientemente grande che rende inattivo il vincolo quando $c_{kt} = 0$. Un tipico valore che le può essere assegnato è $M = n$.

In aggiunta ai vincoli proposti da *Bertsimas e Dunn* sono stati introdotti due ulteriori vincoli che permettono di relazionare le variabili l_t e d_t, rappresentati da:

$$l_t = d_{t/2}, \qquad \forall t \in T_L \mid t \mod 2 == 0, \tag{2.25}$$

$$l_t = d_{a\_l\_max_t}, \qquad \forall t \in T_L \backslash \{T\} \mid t \mod 2 \neq 0, \tag{2.26}$$

dove la variabile $a\_l\_max_t$ rappresenta l'antenato sinistro di valore massimo della foglia t. In particolare il primo vincolo obbliga tutte le foglie di valore pari ad essere attive se al livelo superiore è stato fatto uno *spit* mentre il secondo vincolo rende attive le foglie di valore dispari se è stato effettuato uno *split* nel *Branch Node* antenato sinistro di valore massimo, per la foglia in questione.

Questi vincoli sono stati introdotti in quanto in fase di *cross-validazione* è capitato più volte che ci fosse uno *split* in un *Branch Node* ma poi seguendo il percorso fino alla relativa foglia, questa non fosse attiva o viceversa.

A questo punto si può introdurre la funzione obiettivo, costituita da due contributi:

1. L'errore di classificazione totale: $\sum_{t \in T_L} L_t$, il quale viene normalizzato con l'accuratezza di base $\hat{L}$, ottenuta calcolando il valore più popolare della classe dell'intero dataset, in modo tale da rendere α (parametro usato per gestire la complessità) indipendente dalla numerosità del dataset.

2. La complessità dell'albero, ovvero il numero totale di split: $\sum_{t \in T_B} d_t$.

La funzione obiettivo dunque può essere così scritta:

$$\min \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} d_t. \tag{2.27}$$

Il modello completo è il seguente:

$$\min \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} d_t \tag{2.28}$$

t.c.

$$L_t \geq N_t - N_{kt} - M(1 - c_{kt}), \quad k = 1, .., K, \quad \forall t \in T_L,$$

$$L_t \leq N_t - N_{kt} + M c_{kt}, \quad k = 1, .., K, \quad \forall t \in T_L,$$

$$L_t \geq 0 \quad \forall t \in T_L,$$

$$N_{kt} = \frac{1}{2} \sum_{i=1}^{n} (1 + Y_{ik}) z_{it}, \quad k = 1, .., K, \quad t \in T_L,$$

$$N_t = \sum_{i=1}^{n} z_{it}, \quad \forall t \in T_L,$$

$$\sum_{k=1}^{K} c_{kt} = l_t, \quad \forall t \in T_L,$$

$$\mathbf{a}_m^\top (\mathbf{x}_i + \varepsilon) + < b_m + (1 + \varepsilon_{max})(1 - z_{it}), \ i = 1, .., N, \ \forall t \in \{T_L\}, \ \forall m \in A_L(t),$$

$$\mathbf{a}_m^\top \mathbf{x}_i \geq b_m - M_2(1 - z_{it}), \quad i = 1, .., N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_R(t),$$

$$\sum_{t \in T_L} z_{it} = 1, \quad i = 1, .., N,$$

$$z_{it} \leq l_t, \quad \forall t \in T_L,$$

$$\sum_{i=1}^{n} z_{it} \geq N_{min} l_t, \quad \forall t \in T_L,$$

$$\sum_{j=1}^{p} a_{jt} = d_t, \quad \forall t \in T_B,$$

$$0 \leq b_t \leq d_t, \quad \forall t \in T_B,$$

$$d_t \leq d_p(t), \quad \forall t \in T_B \backslash \{1\},$$

$$l_t = d_{t/2}, \quad \forall t \in T_L \mid t \mod 2 == 0,$$

$$l_t = d_{a\_l\_max_t}, \quad \forall t \in T_L \backslash \{T\} \mid t \mod 2 \neq 0,$$

$$z_{it}, l_t \in \{0,1\}, \quad i = 1,\ldots,N, \quad \forall t \in T_L,$$

$$a_{jt}, d_t \in \{0,1\}, \quad j = 1,..,P, \quad \forall t \in T_B.$$

2.3 Albero decisionale ottimizzato (caso multivariato)

Una naturale estensione del modello appena descritto consiste nel permettere che gli *split* possano coinvolgere più caratteristiche piuttosto che una sola. Le variabili $\mathbf{a}_t$ assolvono ancora la medesima funzione di prima, ma al posto di essere binarie, sono definite nell'intervallo $[-1,1]^p$, ovvero $\mathbf{a}_t \in [-1,1]^p$. Il vincolo (2.3) diventa di conseguenza:

$$\sum_{j=1}^{p} \left| a_{jt} \right| \leq d_t, \quad \forall t \in T_B. \tag{2.29}$$

Il vincolo (2.29) viene linearizzato introducendo le variabili ausiliarie $\hat{a}_{jt}$ e le seguenti disequazioni:

$$\sum_{j=1}^{p} \hat{a}_{jt} \leq d_t, \quad \forall t \in T_B, \tag{2.30}$$

$$\hat{a}_{jt} \geq a_{jt} \quad \forall t \in T_B, \tag{2.31}$$

$$\hat{a}_{jt} \geq -a_{jt} \quad \forall t \in T_B. \tag{2.32}$$

Il successivo vincolo che richiede una modifica è il (2.4) in quanto la quantità $\mathbf{a}^\top \mathbf{x}_i$ non appartiene più all'intervallo $[0,1]^p$ bensì $\mathbf{a}^\top \mathbf{x}_i \in [-1,1]^p$, di conseguenza si introduce:

$$-d_t \leq b_t \leq d_t, \quad \forall t \in T_B. \tag{2.33}$$

La prossima modifica riguarda i vincoli (2.10) e (2.11). Questa deriva dal fatto che la quantità $\mathbf{a}_m^\top \mathbf{x}_i - b_m$ adesso è compresa nell'intervallo $[-2,2]$, di conseguenza $M_1 = M_2 =$

2. Inoltre a differenza del caso univariato non si riesce più a trovare in modo intelligente il parametro ε_j, bensì bisogna sceglierlo in modo arbitrario, per esempio $\varepsilon = \mu = 0.00005$.

In conclusione si possono riscrivere i vincoli come:

$$\mathbf{a}_m^\top \mathbf{x}_i + \mu < b_m + 2(1 - z_{it}), \quad i = 1,..,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t), \tag{2.34}$$

$$\mathbf{a}_m^\top \mathbf{x}_i \geq b_m - 2(1 - z_{it}), \quad i = 1,..,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_R(t). \tag{2.35}$$

Nel regime univariato si penalizzava il numero totale di *split* dell'albero, ora, si penalizza il numero totale di variabili usate nelle divisioni. Per far ciò si introduce la variabile binaria s_{jt}, che vale 1 se il j-esimo attributo è usato nel t-esimo *split*.

Il seguente nuovo vincolo introdotto mostra come la variabile s_{jt} è obbligata ad assumere il valore 1 qualora la j-esima caratteristica sia usata nel nodo t-esimo per splittare:

$$s_{jt} \leq a_{jt} \leq s_{jt}, \quad j = 1,..,P, \quad \forall t \in T_B. \tag{2.36}$$

Il passo successivo è assicurarsi che s_{jt} e d_t siano compatibili, ovvero che d_t possa assumere il valore 1 se e solo se almeno una variabile è usata nello split:

$$s_{jt} \leq d_t \quad j = 1,..,P, \quad \forall t \in T_B, \tag{2.37}$$

$$\sum_{j=1}^{p} s_{jt} \geq d_t, \quad \forall t \in T_B. \tag{2.38}$$

Infine si può scrivere la funzione obiettivo che come nel caso univariato è composta da due contributi. L'unica differenza è che la complessità dell'albero viene misurata dal numero totale di variabili usate nello split, di conseguenza le variabili s_{jt} sostituiranno le d_t, ovvero:

$$\min \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} \sum_{j=1}^{p} s_{jt}. \tag{2.39}$$

Il modello completo è il seguente:

$$\min \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} \sum_{j=1}^{p} s_{jt} \tag{2.40}$$

t.c.

$$L_t \geq N_t - N_{kt} - N(1-c_{kt}), \quad k = 1,..,K, \quad \forall t \in T_L,$$

$$L_t \leq N_t - N_{kt} + Nc_{kt}, \quad k = 1,..,K, \quad \forall t \in T_L,$$

$$L_t \geq 0 \quad \forall t \in T_L,$$

$$N_{kt} = \frac{1}{2}\sum_{i=1}^{n}(1+Y_{ik})z_{it}, \quad k = 1,..,K, \quad t \in T_L,$$

$$N_t = \sum_{i=1}^{n} z_{it}, \quad \forall t \in T_L,$$

$$\sum_{k=1}^{K} c_{kt} = l_t, \quad \forall t \in T_L,$$

$$\mathbf{a}_m^\top \mathbf{x}_i < b_m + M_1(1-z_{it}), \quad i = 1,..,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t),$$

$$\mathbf{a}_m^\top \mathbf{x}_i \geq b_m - M_2(1-z_{it}), \quad i = 1,..,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_R(t),$$

$$\sum_{t \in T_L} z_{it} = 1, \quad i = 1,..,N,$$

$$z_{it} \leq l_t, \quad \forall t \in T_L,$$

$$\sum_{i=1}^{n} z_{it} \geq N_{min} l_t, \quad \forall t \in T_L,$$

$$\sum_{j=1}^{p} \hat{a}_{jt} \leq d_t, \quad \forall t \in T_B,$$

$$\hat{a}_{jt} \geq a_{jt} \quad \forall t \in T_B,$$

$$\hat{a}_{jt} \geq -a_{jt} \quad \forall t \in T_B,$$

$$s_{jt} \leq a_{jt} \leq s_{jt}, \quad j = 1,..,P, \quad \forall t \in T_B,$$

$$s_{jt} \leq d_t \quad j = 1,..,P, \quad \forall t \in T_B,$$

$$\sum_{j=1}^{p} s_{jt} \geq d_t, \quad \forall t \in T_B,$$

$$-d_t \leq b_t \leq d_t, \quad \forall t \in T_B,$$

$$d_t \leq d_p(t), \quad \forall t \in T_B \backslash \{1\},$$

$$l_t = d_{t/2}, \quad \forall t \in T_L \mid t \mod 2 == 0,$$

$$l_t = d_{a\_l\_max_t}, \quad \forall t \in T_L \backslash \{T\} \mid t \mod 2 \neq 0,$$

$$z_{it}, l_t \in \{0,1\}, \quad i = 1,..,N, \quad \forall t \in T_L,$$

$$s_{jt}, d_t \in \{0,1\}, \quad j = 1,..,p, \quad \forall t \in T_B.$$

Capitolo 3

Implementazione dell'Albero decisionale ottimizzato in linguaggio AMPL

In questo capitolo verrà prima introdotto il linguaggio di programmazione AMPL, successivamente verrà descritto come il modello illustrato nel capitolo precedente sia stato implementato in questo linguaggio di ottimizzazione.

3.1 Il linguaggio di programmazione matematica AMPL

AMPL [5] è un linguaggio di modellazzione algebrica implementato dai creatori Robert Fourer, David Gay e Brian W. Kernighan, nasce nel 1985 nei Bell Laboratories con l'obiettivo di aiutare gli utenti nella formulazione di modelli di programmazione matematica.

L'esigenza della sua creazione nasce dal fatto che le modalità espressive del linguaggio di programmazione e dell' algoritmo stesso sono differenti, questo significa che scrivere un algoritmo per la risoluzione di un determinato problema e scrivere il programma per far eseguire l'algoritmo sono due processi distinti e tipicamente questa traduzione comporta un notevole dispendio di tempo, nonché l'efficienza dell'algoritmo può risultare peggiore delle aspettative in quanto vincolato alla sintassi del linguaggio.

AMPL si propone come un nuovo strumento per facilitare l'attività di modellazione e renderla meno soggetta ad errori. Sinteticamente, si può riassumere la forza di questo linguaggio nei seguenti punti:

- sfrutta una sintassi molto simile alla notazione algebrica-simbolica, comunemente utilizzata per descrivere modelli matematici e allo stesso tempo preserva una struttura sufficientemente formale per essere processata su qualsiasi programma di ottimizzazione, anche chiamato solver.

- mantiene la struttura logica del modello (descritto con variabili, parametri, insiemi, vincoli e funzioni obiettivo) autonoma rispetto al valore delle specifiche istanze.

- possiede un "traduttore" implementato al suo interno che, presi come input i modelli e i dati associati, produce un output per i più importanti ottimizzatori a disposizione sul mercato, dal momento che ognuno di essi propone un proprio ambiente integrato di sviluppo dei modelli e strutture dati.

- garantisce al modello completa indipendenza dai diversi risolutori e allo stesso tempo permette all'utente di passare velocemente da un solver all'altro, confrontandone le prestazioni in funzione del modello utilizzato.

L'insieme di queste caratteristiche rendono AMPL uno strumento molto versatile, poiché permette all'utente di approcciarsi con un unico linguaggio a diversi ottimizzatori quando invece ognuno di questi avrebbe richiesto una particolare tecnica di programmazione.

Quindi al tempo stesso facilita enormemente la formulazione di modelli di ottimizzazione e genera strutture dati con i corretti requisiti computazionali. I principali ottimizzatori disponibili su AMPL si possono suddividere in due macro categorie, in funzione delle proprietà del modello studiato:

- CPLEX, Gurobi, Xpress sono solver adatti per la risoluzione di modelli lineari o quadratici, con variabili intere, continue o miste.
- KNITRO, MINOS, SNOPT permettono di trovare soluzioni ottime locali a problemi non lineari con variabili continue.

In particolare per lo svolgimento di questo lavoro di tesi è stato scelto il risolutore Gurobi il quale è capace di risolvere problemi di programmazione lineare intera e mista e programmazione quadratica. La scelta è ricaduta su tale strumento in quanto è ritenuto uno dei risolutori più efficienti ed efficaci sul mercato, in grado di funzionare con i più svariati linguaggi di programmazione e modellazione: Python, Java C++, AIMMS, AMPL, MatLab e molti altri.

Per ultimo va detto che il funzionamento di qualsiasi programma in AMPL è basato sulla presenza di tre file, distinti da altrettante estensioni:

1. il file *.mod* dove si sviluppa la struttura logica del modello e contiene le dichiarazioni di insiemi, parametri, variabili, vincoli e funzione obiettivo.
2. il file *.dat* dove vengono salvati i dati specifici dei modelli, lasciando così a questi ultimi un certo grado di indipendenza.
3. il file *.run* riporta le informazioni relative al risolutore scelto, il richiamo dei file *.mod* e *.dat* necessari alla risoluzione e l'enunciazione di ciò che il risolutore deve calcolare e riportare nella finestra di dialogo.

Questi file verranno analizzati più approfonditamente nella sezione successiva.

3.2 Il modello OCT in AMLP

Segue ora una descrizione dettagliata del contenuto dei tre file esposti nel paragrafo precedente, il primo a essere esaminato è il *.mod* che è il file principale in quanto contiene il modello di ottimizzazione.

Le entità principali di questo file sono: gli insiemi, i parametri, le variabili, i vincoli e la funzione obiettivo.

Tipicamente in un modello di programmazione matematica il primo aspetto da definire sono gli insiemi ovvero una collezione di oggetti la cui funzione principale è quella di indicizzare variabili e vincoli, ossia di riferirsi al contenuto di quest' ultimi tramite un indice che è rappresentato da uno degli elementi dell'insieme considerato.

Poiché gli insiemi sono così fondamentali, AMPL offre un'ampia varietà di tipi di insiemi e di operazioni. I membri di un insieme possono essere stringhe o numeri, ordinati o non ordinati; possono presentarsi singolarmente o come coppie, triple o "tuple" più lunghe. Gli insiemi possono essere definiti elencando o calcolando esplicitamente i loro membri oppure applicando operazioni come unione e intersezione da altri insiemi.

La sintassi necessaria per dichiarare un insieme è:

```
set nome_insieme proprietà_insieme;
```

dove *set* è la parola chiave necessaria per introdurre la dichiarazione, si noti che in AMPL ogni istruzione è terminata dal ";". Per questo modello sono stati definiti i seguenti insiemi:

```
set OSSERVAZIONI;
```

Rappresenta l'insieme che contiene il numero progressivo di identificazione del paziente, in questo caso da 1 a 150.

```
set FEATURES;
```

Rappresenta l'insieme che contiene il numero progressivo di identificazione delle caratteristiche, in questo caso da 1 a 15. In particolare si ha la seguente associazione:

1. **Test COVID-19**

2. **Age**
3. **Cronic Diseases**
4. **Diabetes**
5. **Respiratory Diseases**
6. **Kidney Diseases**
7. **Neutr**
8. **D-dimero**
9. **AST**
10. **LDH**
11. **CK**
12. **PCR**
13. **cTnI**
14. **P/F**
15. **WBC/Linf**

`set BRANCH_NODES;`

Rappresenta l'insieme contenente i nodi non foglia.

`set LEAF_NODES;`

Rappresenta l'insieme contenente i nodi foglia.

`set TOTAL_NODES;`

Rappresenta l'insieme contenente i nodi totali dell'albero.

`set CLASSI;`

Rappresenta l'insieme delle classi.

```
set A_L {TOTAL_NODES};
```

Rappresenta l'insieme degli antenati sinistri di un determinato nodo, infatti come si può vedere, a differenza degli insiemi sopra elencati, l'insieme A_L è definito sull'insieme $TOTAL\_NODES$, cioè per ogni membro di tale insieme è definito l'insieme degli antenati sinistri.

```
set A_R {TOTAL_NODES};
```

Rappresenta l'insieme degli antenati destri di un determinato nodo.

Vengono ora presentati i parametri usati dal modello, ovvero quei valori costanti che vengono forniti in input al modello. La sintassi necessaria per dichiararli è:

```
param nome_parametro proprietà_parametro;
```

Come per le variabili, se necessario, è possibile indicizzare un determinato parametro attraverso gli insiemi sopra definiti, come verrà fatto per i parametri x, y e Y.

```
param N;
```

Rappresenta il numero di osservazioni.

```
param P;
```

Rappresenta il numero di attributi.

```
param T;
```

Rappresenta il numero totale di nodi dell'albero.

```
param D;
```

Rappresenta l'altezza dell'albero, verrà spiegato successivamente com'è stato calcolato.

```
param N_min;
```

Rappresenta il numero minimo di osservazioni che possono cadere in una foglia, verrà spiegato successivamente com'è stato calcolato.

```
param mu;
```

Parametro usato per eliminare la disuguaglianza stretta nelle equazioni che impongono la struttura dell'albero.

```
param alpha;
```

Parametro usato per controllare la complessità dell'albero.

```
param L_tilda;
```

Parametro usato per normalizzare l'errore di classificazione.

```
param temp;
```

Parametro usato per calcolare gli insiemi degli antenati sinistri e destri.

```
param x {OSSERVAZIONI, FEATURES};
```

Matrice che contiene per ogni paziente i dati relativi ai parametri laboratoristici.

```
param y {OSSERVAZIONI} binary;
```

Vettore che contiene per ogni paziente la classe di appartenenza.

```
param Y {i in OSSERVAZIONI, k in CLASSI} = if y[i] == k
   then 1 else -1;
```

Parametro che rappresenta l'antenato sinistro di valore massimo di una foglia.

```
param a_l_max {t in LEAF_NODES diff {15}: t mod 2 != 0};
```

Matrice usata per memorizzare le classi delle 150 osservazioni.

Vengono ora presentate le variabili usate dal modello, ovvero quei valori che sarà compito del solutore calcolare e che rappresenteranno la soluzione al problema precedentemente esposto. La sintassi necessaria per dichiararle è:

```
var  nome_variabile proprietà_variabile;
```

Si noti inoltre come per alcune variabili si è deciso di definirle direttamente durante la fase di dichiarazione, questo per ridurre il numero di vincoli del modello e quindi dare una migliore leggibilità a quest'ultimo, un esempio di ciò è espresso dalle variabili N_{kt} e N_t.

Per ultimo si osservi come anche alcuni vincoli che devono soddisfare le variabili possono essere definiti direttamente durante la loro dichiarazione, come nel caso della variabile **a** che è definita all'interno dell'intervallo $[-1, 1]$.

```
var z {OSSERVAZIONI, TOTAL_NODES} binary ;
```

Memorizza per ogni nodo le osservazioni che vi appartengono.

```
var l {LEAF_NODES} binary;
```

Usata per memorizzare lo stato di una foglia: attiva/disattiva.

```
var s {FEATURES, BRANCH_NODES} binary;
```

Ha il compito di memorizzare le caratteristiche usate negli split.

```
var d {BRANCH_NODES} binary;
```

Usata per sapere se un determinato *Branch Node* splitta o meno.

```
var b {BRANCH_NODES};
```

Variabile usata come soglia negli split.

```
var a {FEATURES, BRANCH_NODES} <= 1 , >= -1;
```

Variabile usata per selezionare le caratteristiche che partecipano allo split ad un determinato *Branch Node*.

```
var a_capp {FEATURES, BRANCH_NODES} <= 1 , >= -1;
```

Variabili introdotte per linearizzare il valore assoluto del vincolo (2.29).

```
var N_kt {k in CLASSI,t in LEAF_NODES} = 1/2 * sum {i in
   OSSERVAZIONI}(1 + Y[i,k])*z[i,t];
```

Variabile usata per memorizzare il numero di osservazioni con una determinata etichetta che appartengono ad un determinato nodo.

```
var N_t {t in LEAF_NODES} = sum {i in OSSERVAZIONI} z[i,t];
```

Memorizza il numero totale di osservazioni in un determinato nodo foglia.

```
var c_kt {k in CLASSI, t in LEAF_NODES} binary;
```

Usata per memorizzare le classi dei nodi foglia.

```
var L_t{t in LEAF_NODES} >=0;
```

Variabile che rappresenta l'errore di classificazione.

Arrivati a questo punto le ultime due entità da definire sono la funzione obiettivo e i vincoli.

La funzione obiettivo di un problema di minimizzazione è preceduta dalla funzione AMPL *"minimize"*, dopodiché è necessario darle un nome che in questo caso molto semplicemente è *"obiettivo"*.

```
minimize obiettivo:
  sum {t in LEAF_NODES} (1/L_tilda) * L_t[t]
    + alpha * sum{t in BRANCH_NODES, j in FEATURES}
       s[j,t];
```

I vincoli sono invece le condizioni che la soluzione deve soddisfare e rappresentano la regione ammissibile, anch'essi necessitano di essere convertiti in linguaggio AMPL. Per fare ciò la dichiarazione di ogni vincolo inizia con *"s.t."* o *"subject to"* che dall'inglese significa *"tale che"*, dopodiché va scelto un nome e vanno riportati gli indici su cui il modello dovrà lavorare per la risoluzione.

```
subject to eq1 {t in BRANCH_NODES diff {1}}:
  d[t] <= d[floor(t/2)];
```

Rappresenta il vincolo: $d_t \leq d_{p(t)}, \quad \forall t \in T_B \backslash \{1\}$.

```
subject to eq2_1 {t in BRANCH_NODES}:
  b[t] <= d[t];

subject to eq2_2 {t in BRANCH_NODES}:
  -d[t] <= b[t];
```

Rappresentano il vincolo: $-d_t \leq b_t \leq d_t, \quad \forall t \in T_B$.

Si noti come non è stato possibile scriverlo in un'unica equazione come da modello bensì è stato necessario uno sdoppiamento in quanto AMPL non permette questa notazione.

```
subject to eq3_1 {t in BRANCH_NODES}:
  sum{j in FEATURES} s[j,t] >= d[t];
```

Rappresenta il vincolo: $\sum_{j=1}^{p} s_{jt} \geq d_t, \quad \forall t \in T_B$.

```
subject to eq3_2 {j in FEATURES, t in BRANCH_NODES}:
  s[j,t] <= d[t];
```

Rappresenta il vincolo: $s_{jt} \leq d_t, \quad j = 1,..,P, \quad \forall t \in T_B$.

```
subject to eq4_1 {j in FEATURES, t in BRANCH_NODES}:
  a[j,t] <= s[j,t];
```

```
subject to eq4_2 {j in FEATURES, t in BRANCH_NODES}:
  -s[j,t] <= a[j,t];
```

Rappresentano il vincolo: $-s_{jt} \leq a_{jt} \leq s_{jt}, \quad j = 1,..,P, \quad \forall t \in T_B$. Come sopra si noti che è stata necessaria un'operazione di sdoppiamento del vincolo.

```
subject to eq5_1 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] >= a[j,t];
```

Rappresenta il vincolo: $\hat{a}_{jt} \geq a_{jt} \quad \forall t \in T_B$.

```
subject to eq5_2 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] >= -a[j,t];
```

Rappresenta il vincolo: $\hat{a}_{jt} \geq -a_{jt} \quad \forall t \in T_B$.

```
subject to eq5_3 {t in BRANCH_NODES}:
  sum {j in FEATURES} a_capp[j,t] <= d[t];
```

Rappresenta il vincolo: $\sum_{j=1}^{p} \hat{a}_{jt} \leq d_t, \quad \forall t \in T_B$.

```
subject to eq6_1{t in LEAF_NODES}:
```

```
  sum {i in OSSERVAZIONI} z[i,t] >= N_min*l[t];
```

Rappresenta il vincolo: $\sum_{i=1}^{n} z_{it} \geq N_{min} l_t, \quad \forall t \in T_L$.

```
subject to eq6_2{i in OSSERVAZIONI, t in LEAF_NODES}:
  z[i,t] <= l[t];
```

Rappresenta il vincolo: $z_{it} \leq l_t, \quad \forall t \in T_L$.

```
subject to eq7 {i in OSSERVAZIONI}:
  sum {t in LEAF_NODES} z[i,t] = 1;
```

Rappresenta il vincolo: $\sum_{t \in T_L} z_{it} = 1, \quad i = 1,..,N$.

```
subject to eq8_1 {i in OSSERVAZIONI, t in TOTAL_NODES, a_l
   in A_L[t]}:
  sum {j in FEATURES} a[j,a_l]*x[i,j] + mu <= b[a_l] + (2
     + mu)*(1 - z[i,t]);
```

Rappresenta il vincolo: $\mathbf{a}_m^\top \mathbf{x}_i + \mu < b_m + M_1(1 - z_{it}), \quad i = 1,..,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t)$.

```
subject to eq8_2 {i in OSSERVAZIONI, t in TOTAL_NODES, a_r
   in A_R[t]}:
  sum {j in FEATURES} a[j,a_r]*x[i,j] >= b[a_r] - 2*(1 -
     z[i,t]);
```

Rappresenta il vincolo: $\mathbf{a}_m^\top \mathbf{x}_i \geq b_m - M_2(1 - z_{it}), \quad i = 1,..,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_R(t)$.

```
subject to eq9 {t in LEAF_NODES}:
  sum {k in CLASSI} c_kt[k,t] = l[t];
```

Rappresenta il vincolo: $\sum_{k=1}^{K} c_{kt} = l_t, \quad \forall t \in T_L$.

```
subject to eq10 {k in CLASSI, t in LEAF_NODES}:
  L_t[t] <= N_t[t] - N_kt[k,t] + N*c_kt[k,t];
```

Rappresenta il vincolo: $L_t \leq N_t - N_{kt} + N c_{kt}, \quad k = 1,..,K, \quad \forall t \in T_L$.

```
subject to eq11 {k in CLASSI, t in LEAF_NODES}:
  L_t[t] >= N_t[t] - N_kt[k,t] - N*(1 - c_kt[k,t]);
```

Rappresenta il vincolo: $L_t \geq N_t - N_{kt} - N(1 - c_{kt}), \quad k = 1,..,K, \quad \forall t \in T_L.$

```
subject to eq12_1 {t in LEAF_NODES: t mod 2 == 0}:
  l[t] = d[t/2];
```

Rappresenta il vincolo: $l_t = d_{t/2}, \quad \forall t \in T_L \mid t \mod 2 == 0$

```
subject to eq12_4 {t in LEAF_NODES diff {T} : t mod 2 !=
  0}:
  l[t] = d[a_l_max[t]];
```

Rappresenta il vincolo: $l_t = d_{a\_l\_max_t}, \quad \forall t \in T_L \backslash \{T\} \mid t \mod 2 \neq 0$

Una volta terminata la descrizione del file *.mod* è necessario discutere il file *.dat*.

Come accennato in precedenza, contiene i dati specifici per una particolare istanza di problema in modo tale da poter separare il modello dai dati garantendo così l'indipendenza tra i due. Tipicamente la sintassi per l'assegnamento di un dato è la seguente:

```
entità nome_entità := valore;
```

oppure

```
let nome_entità := valore;
```

dove per *"entità"* si intendono parametri o insiemi, di conseguenza la parola chiave *"entità"* sarà sostituita dalle parole chiave *"param"* o *"set"*, *nome_entità* rappresenta il nome di quel particolare parametro/insieme mentre *"let"* identifica la parola chiave introduttiva per eseguire un aggiornamento.

La differenza sostanziale tra le due sintassi è che nella prima il membro *"valore"* deve essere una costante e non può essere un dato calcolato a partire da altri parametri ovvero calcolato attraverso un'espressione, di conseguenza il parametro T e gli insiemi *BRANCH_NODES* e *LEAF_NODES* possono essere definiti solo con la seconda espressione e per mantenere un'unica notazione si è quindi scelto di usare sempre la seconda. Per quanto riguarda gli insiemi, sono stati inizializzati nel seguente modo:

```
let OSSERVAZIONI := 1 .. N;
let FEATURES := 1 .. P;
let BRANCH_NODES := 1..floor(T/2);
let LEAF_NODES := floor(T/2) +1 .. T;
let TOTAL_NODES := BRANCH_NODES union LEAF_NODES;
let CLASSI := {0,1};

for{t in TOTAL_NODES}{
let A_L[t] := {} ;
let A_R[t] := {} ;
}

for{t in TOTAL_NODES diff {1} }{
let temp := t;
  repeat {
    if temp mod 2 == 0 then
      let A_L[t] := A_L[t] union {temp/2};
      else
      let A_R[t] := A_R[t] union {floor(temp/2)};
    let temp := floor(temp/2);
    } while temp/2 >=1 ;
}
```

dove la notazione “x .. y” permette di selezionare tutti gli elementi compresi tra x e y inclusi in modo tale evitare un elenco esplicito dei suddetti valori. I parametri del modello sono stati inizializzati nel seguente modo:

```
let N := 150;
let P := 15;
let mu := 0.00005;
let alpha := 0.02;
let D := 3;
```

```
let N_min := 5;
let T := 2^(D+1) -1;

read {i in OSSERVAZIONI, j in FEATURES} x[i,j] < x.txt;
read {i in OSSERVAZIONI} y[i] < y.txt;


for {i in OSSERVAZIONI, j in FEATURES}{
 let x[i,j] := (x[i,j] - min {m in OSSERVAZIONI} x[m,j]) /
    (max {m in OSSERVAZIONI} x[m,j] - min {m in
    OSSERVAZIONI} x[m,j] );
}

if sum {i in OSSERVAZIONI : y[i] == 0} y[i] >= sum {i in
   OSSERVAZIONI : y[i] == 1} y[i] then
  let L_tilda := sum {i in OSSERVAZIONI : y[i] == 0} y[i];
  else
  let L_tilda := sum {i in OSSERVAZIONI : y[i] == 1} y[i];

for{t in LEAF_NODES diff {T}: t mod 2 != 0}{
  let a_l_max[t] := max{m in A_L[t]} m;
}
```

Per quanto riguarda i dati delle osservazioni è stato prima necessario un lavoro di normalizzazione in quanto originariamente non rispettavano la condizione $\boldsymbol{x}_i \in [0,1]^p$, in particolare è stato usato il metodo $min - max$ il quale viene così definito:

$$x = \frac{x - min(x)}{max(x) - min(x)}. \tag{3.1}$$

Per ultimo va precisato che i parametri D, α e N_{min} sono stati calcolati per mezzo di un processo di *tuning* ovvero di ricerca del valore ottimale attraverso un lavoro di addestramento. In particolare per svolgere questa fase la funzione obiettivo è stata riscritta

nel seguente modo:

```
minimize obiettivo:
   sum {t in LEAF_NODES} (1/L_tilda) * L_t[t];
```

ed è stato aggiunto il vincolo:

```
subject to eq0:
   sum {t in BRANCH_NODES, j in FEATURES} s[j,t] <= C;
```

che vede l'aggiunta di un ulteriore parametro C il quale è il limite superiore al numero di variabili usate negli split.

Questo è stato necessario perché originariamente α è un parametro continuo, di conseguenza bisogna discretizzarlo e risolvere il modello per ciascuno dei suoi diversi valori. Questa procedura risulta però essere troppo costosa. Riscrivendo invece la funzione obiettivo e i vincoli nel modo seguente, si può notare che le soluzioni saranno da ricercare nel range $C = 1 \dots C_{max}$ dove $C_{max} = (2^D - 1) * 15$ il quale rappresenta il caso in cui ad ogni *Branch Node* vengono usate tutte le variabili per eseguire l'operazione di split.

La fase di *tuning* ha prodotto i seguenti risultati:

- $C \geq 6$ garantisce l'ottimalità della funzione obiettivo.
- $D \geq 2$ garantisce l'ottimalità della funzione obiettivo purchè $C \geq 6$
- $N_{min} \leq 20$ garantisce l'ottimalità della funzione obiettivo purchè $C \geq 6$.

Sono state quindi tralle le seguenti conclusioni:

- $D = 3$ in quanto teoricamente si preferiscano alberi meno articolati perchè permettono una miglior leggibilità e hanno migliori proprietà di generalizzazione.
- $N_{min} = 5$ rappresenta il giusto compromesso tra interpretabilità e generalizzazione. Scegliere un N_{min} più piccolo significa d'altro canto avere un albero molto frammentato con foglie che contengono 1 o pochi punti con conseguente *over fitting* e difficoltà di interpretazione. Al contrario un N_{min} elevato obbligherebbe l'algoritmo a inserire punti in foglie non idonee solo perché è costretto dal vincolo e di

conseguenza si avrebbe un comportamento non corretto, visibile in un errore di classificazione elevato.

- $\alpha = 0.02$. Si è cercato il valore ottimale di α nel range $[\frac{1}{(2^D-1)\cdot 15} \dots \frac{1}{C}]$ ottenendo così il valore 0.045; si è però deciso di portare questo valore a 0.02 in quanto in fase di *cross-validazione* è stato quello che ha permesso di ottenere migliori risultati in termini di capacità di generalizzazione del modello. Questo risultato è dovuto al fatto che nella fase di *tuning* per velocizzare il processo è stato usato il *Validation set approach* che è metodo che soffre maggiormente di come i dati sono stati suddivisi in *Training Set* e *Test Set* rispetto alla *K-fold cross validation*, metodo uato nella *cross-validazione*.

Infine viene presentato il file *.run*. Questo file ha il compito di richiamare i file *.mod* e *.dat* in modo da fornirli al risolutore. A tal fine è buona pratica iniziare tale file con il comando *"reset"* così da assicurarsi che il risolutore parta da zero in quanto potrebbero esserci attività in corso che potrebbero inficiare l'efficienza dello stesso. I file *.mod* e *.dat* vengono caricati con i comandi *model* e *data*, il comando *"option solver"* seguito dal nome del risolutore consente di sceglierlo, viene poi indicato il tempo limite impostato a 1000 secondi. Il comando *"solve"* dà inizio al processo risolutivo. Terminato ciò, compare sulla console AMPL il messaggio con la soluzione, seguita dalle variabili e la funzione obiettivo indicate con il comando *"display"*. Per richiamare il file *.run* è necessario digitare *include init.run* all'interno della console AMPL.

L'intero codice è riportato in appendice.

Capitolo 4

Validazione del modello

In questo capitolo verranno introdotti i principali indicatori per studiare la bontà del modello e successivamente verrà illustrato il metodo usato per la validazione.

4.1 Testing del modello

Una volta scritto e tradotto il modello in AMPL è necessario studiarne la sua affidabilità ed efficacia mediante specifici indicatori, in particolare quelli che verranno trattati sono: accuratezza, richiamo e precisione.

Questi permetteranno di trarre delle conclusioni sul modello in modo tale da capire se effettivamente può essere usato per fare delle predizioni affidabili su dati nuovi. Prima di entrare nel dettaglio del calcolo di questi parametri si descriverà la fase di allenamento e validazione.

Le tre alternative disponibili sono:

1. *Validation set approach*: consiste nel dividere in modo casuale l'insieme di dati disponibili in due parti ovvero *training set* e *test set* e di conseguenza allenare il modello sul primo insieme e testarlo sul secondo.

2. *Leave-one-out cross validation (LOOCV)* : divide il set di osservazioni in due parti, tuttavia, invece di creare due sottoinsiemi di dimensioni paragonabili, procede come segue:

    - una singola osservazione è utilizzata per la validazione e le restanti osservazioni compongono l'insieme di addestramento.
    - Le variabili del modello sono stimate sulla base delle osservazioni del *training set.*
    - La predizione viene fatta sulla base della singola osservazione.
    - La procedura viene iterata n (numero delle osservazioni) volte, dove ad ogni iterazione, viene presa come singola osservazione una diversa da quelle precedentemente considerate.

3. *Validazione incrociata K-fold cross validation*: in questo approccio si divide l'insieme delle N osservazioni in k gruppi o *folders* ognuno di dimensione h. Successivamente una *folder* per volta viene usata come *test set* e le $k-1$ restanti vengono usate come *training set.* Questa procedura viene iterata esattamente k volte, come illustrato in figura 4.1.

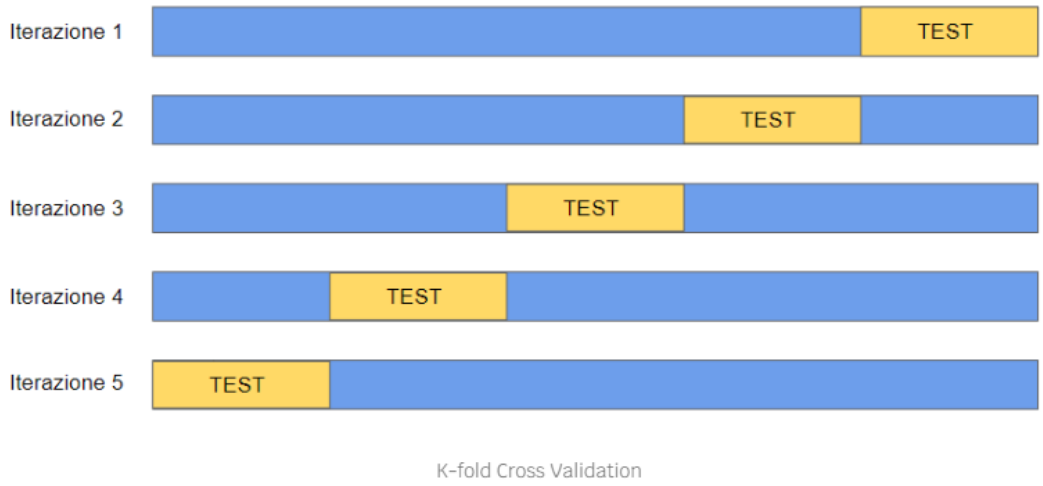

Figura 4.1: Esempio grafico K-fold cross validation. (con k=5)

Fonte: Altervista.org [1]

Considerate le 3 alternative la scelta è caduta sulla *K-fold cross validation* in quanto è quella che permetteva di avere meno variabilità nei risultati con un dispendio computazionale discreto. Infatti il *validation set approach* ha il difetto dell'elevata variabilità, ovvero i risultati possono subire sostanziali cambiamenti a seconda di come vengono composti gli insiemi oltre al fatto che solo una parte delle unità disponibili sono usate per la stima, potendo così provocare una minor precisione stessa. Gli svantaggi del *leave-one-out cross validation* sono invece una stima potenzialmente scadente, in quanto fatta su una singola osservazione e un dispendio computazionale oneroso.

Una volta scelto il metodo di convalida bisogna procedere al suo settaggio. In particolare a seguito di un'analisi di sensitività rispetto alle diverse cardinalità delle *folders* si è giunti alla conclusione che una cardinalità più bassa genera risultati più precisi infatti è maggiore la quantità di dati alla quale può attingere il modello per allenarsi.

In virtù di ciò è stato scelto $k = 50$, ovvero i dati sono stati suddivisi in 50 sottoinsiemi da 3 osservazioni l'una, di conseguenza, in ognuna delle 50 iterazioni il modello è stato addestrato con 147 dati e poi convalidato con i restanti 3.

Una volta descritto il funzionamento del metodo di validazione incrociata *K-fold* viene ora presentato l'effettivo calcolo dei parametri di cui sopra, in particolare si ha:

$$\%Accuratezza_i = \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i} 100, \quad i = 1, \ldots, 50,$$

$$\%Richiamo_i = \frac{TP_i}{TP_i + FN_i} 100, \quad i = 1, \ldots, 50,$$

$$\%Precisione_i = \frac{TP_i}{TP_i + FP_i} 100, \quad i = 1, \ldots, 50, \tag{4.1}$$

dove TP_i rappresenta il numero di veri-positivi (*true positive*), TN_i dei veri-negativi (*true negative*), FP_i dei falsi-positivi (*false positive*) e FN_i dei falsi-negativi (*false negative*) del sottoinsieme i. Viene poi calcolato un punteggio medio per ciascun indicatore:

$$\%Accuratezza = \frac{1}{k} \sum_{i=1}^{k} \%Accuratezza_i,$$

$$\%Richiamo = \frac{1}{k} \sum_{i=1}^{k} \%Richiamo_i,$$

$$\%Precisione = \frac{1}{k} \sum_{i=1}^{k} \%Precisione_i. \tag{4.2}$$

4.2 Validazione del modello in AMPL

Al fine di automatizzare la fase di addestramento e validazione è stato creato un apposito file denominato *"init.run"* in cui vengono eseguite in modo sequenziale queste due attività per un totale di 50 volte.

In particolare in ogni iterazione, dopo aver addestrato il modello su 147 dati, vengono usate le variabili decisionali prodotte al fine di effettuare una previsione sulle osservazioni del *set* di validazione.

Per far ciò vengono usati due cicli *for*, il primo, date le variabili decisionali **a** e b, ha il compito di calcolare il nodo foglia in cui cade un'osservazione:

```
param nodo_foglia_calcolato {OSSERVAZIONI_VAL};
```

```
for{i in OSSERVAZIONI_VAL}{
     let t_val := 1;

     repeat {

          let temp_val := 0.0;

          for{p in FEATURES}{
            let temp_val := temp_val + a[p, t_val] *
               x_val[i,p];
          }

          if temp_val < b[t_val] then{
            let nodo_foglia_calcolato[i] := t_val*2;
            let t_val := t_val*2;
          }else{
            let nodo_foglia_calcolato[i] := t_val*2 + 1;
            let t_val := t_val*2 + 1;
          }

        }while t_val <= floor(T/2) ;
   }
```

mentre il secondo serve a calcolare la classe di appartenenza di un'osservazione sulla base del nodo foglia assegnatogli precedentemente:

```
param classe_calcolata {OSSERVAZIONI_VAL};

for{i in OSSERVAZIONI_VAL}{
   for{l_val in LEAF_NODES}{
```

```
        if c_kt[0, l_val] == 1 then{
            if nodo_foglia_calcolato[i] == l_val then
                let classe_calcolata[i] := 0;
        }

        if c_kt[1, l_val] == 1 then{
            if nodo_foglia_calcolato[i] == l_val then
                let classe_calcolata[i] := 1;
        }
    }
}
```

Una volta ottenuto il vettore *classe_calcolata* si sono potuti calcolare i veri positivi, veri negativi, falsi positivi e falsi negativi:

```
    for{i in OSSERVAZIONI_VAL}{

      if classe_calcolata[i] == 0 and y_val[i] == 0 then
        let veri_negativi := veri_negativi + 1;

      if classe_calcolata[i] == 1 and y_val[i] == 1 then
        let veri_positivi := veri_positivi + 1;

      if classe_calcolata[i] == 0 and y_val[i] == 1 then
        let falsi_negativi := falsi_negativi + 1;

      if classe_calcolata[i] == 1 and y_val[i] == 0 then
        let falsi_positivi := falsi_positivi + 1;
    }
```

Infine vengono calcolati gli indicatori del modello con le formule esposte precedentemente.

Il modello è stato addestrato su un computer ASUS a 64 bit con 8 GB di RAM e un

processore Intel i5-6200u da 2.8 GHz. Ad ogni istanza è stato fornito un tempo massimo di 1200 secondi, di conseguenza il tempo complessivo utilizzzato per la fase di addestramento è stato circa di 16 ore e 38 minuti.

Capitolo 5

Risultati computazionali

5.1 Modello Univariato

5.1.1 Struttura dell'albero

Il modello è stato allenato utilizzando i dati descritti nel capitolo 1 e validato come descritto nel capitolo 4.

Le seguenti tabelle riportano i risultati ottenuti in AMPL:

a [*,*]:	1	2	3	4	5	6	7
covid test	0	0	0	0	0	0	0
age	0	0	0	0	0	0	0
cronic	0	0	0	0	0	0	0
diabets	0	0	0	0	0	0	0
respiratory	0	0	0	0	0	0	0
kidney	0	0	0	0	0	0	0
neutr	0	0	0	0	0	0	0
d-dimero	0	0	0	0	0	0	0
AST	0	0	0	1	0	0	0
LDH	0	1	1	0	0	0	0
CK	0	0	0	0	0	0	0
PCR	0	0	0	0	0	0	0
cTnL	1	0	0	0	0	0	0
WBC/Linf	0	0	0	0	0	0	0
P/F	0	0	0	0	0	1	0

Tabella 5.1: Soluzione relativa alla variabile $\mathbf{a}_t$.

b[*]:	
1	0.00195417
2	0.508621
3	0.212644
4	0.0291312
5	0
6	0.331941
7	0

Tabella 5.2: Soluzione relativa alla variabile b.

L_t[*]:	
8	1
9	8
10	0
11	1
12	0
13	1
14	0
15	8

Tabella 5.3: Soluzione relativa alla variabile L_t.

c_kt:		
0	8	0
0	9	1
0	10	0
0	11	0
0	12	0
0	13	1
0	14	0
0	15	0
1	8	1
1	9	0
1	10	0
1	11	1
1	12	1
1	13	0
1	14	0
1	15	1

Tabella 5.4: Soluzione relativa alla variabile c_{kt}.

I quali hanno permesso la costruzione del seguente albero:

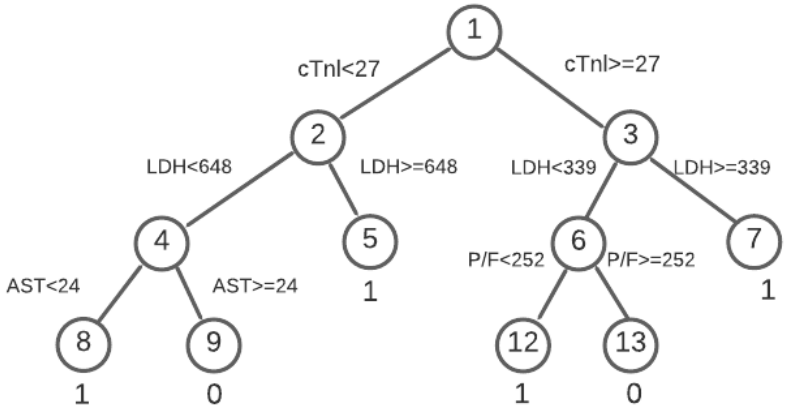

Figura 5.1: Struttura finale albero modello univariato.

Come si può notare dal grafico 5.1 solo 4 dei 15 attributi totali sono stati usati per la sua rappresentazione, i quali identificano quindi le variabili di maggior importanza per determinare il decorso di un paziente malato di covid. C'è da sottolineare che questo risultato è coerente con i risultati del test chi-quadro (1.1) esposti nel capitolo 1 infatti cTnL, LDH, e P/F rientrano tra i 5 attributi a maggior potere predittivo selezionati inizialmente.

La lettura del grafico 5.1 è molto semplice, infatti partendo dal nodo radice, sulla base dei parametri vitali e laboratoristici di un paziente bisogna seguire il percorso fino a giungere ad una determinata foglia la quale indicherà il decorso del paziente, ricordando che la classe 1 indica una scarsa probabilità di sopravvivenza mentre la 0 identifica la probabilità contraria. Per esempio se un paziente avesse il valore della Troponina cardiaca (cTnL) < 27 e lattato deidrogenasi (LDH) ≥ 648 allora la sua probabilità di sopravvivenza sarebbe bassa, al contrario se avesse un valore di lattato deidrogenasi < 648, pur restando con un valore di Troponina cardiaca < 27 bisognerebbe valutare l'AST che se risultasse ≥ 24 indicherebbe un decorso positivo per il malato. Tali risultati concordano con quanto presentato in [6, 10].

5.1.2 Proprietà dell'albero

Una volta presentata la struttura dell'albero finale è possibile quantificare le sue proprietà, in particolare per l'albero illustrato in figura 5.1 si è riscontrata una *in-sample accuracy* dell' **88.6%**, ovvero è stato in grado di classificare correttamente l' **88.6%** dei dati fornitegli nella fase di allenamento. In particolare la figura 5.2 rappresenta la *matrice di confusione* dell'albero in questione: sull'asse delle ordinate si trovano le vere classi delle osservazioni mentre sull'asse delle ascisse si trovano le classi predette dal modello.

Incrociando una riga con una determinata colonna si trova la percentuale di classi predette correttamente o al contrario la percentuale di quelle predette erroneamente. Per esempio il quadrante alla prima riga e prima colonna identifica la percentuale di pazienti classificati correttamente come sopravvissuti, infatti la *true class* e la *predicted class* sono entrambe uguali a zero.

Il quadrante alla seconda riga e prima colonna identifica invece la percentuale di pazienti classificati erroneamente come sopravvissuti, infatti la *true class* è uguale a uno mentre la *predicted class* è uguale a 0. Ragionamenti analoghi si possono fare per gli altri quadranti.

Queste proprietà forniscono una prima valutazione sommaria dell'albero in quanto avere dei buoni valori per l' *in-sample accuracy* è solo una condizione necessaria per il corretto funzionamento dell'algoritmo. La vera potenza predittiva del modello

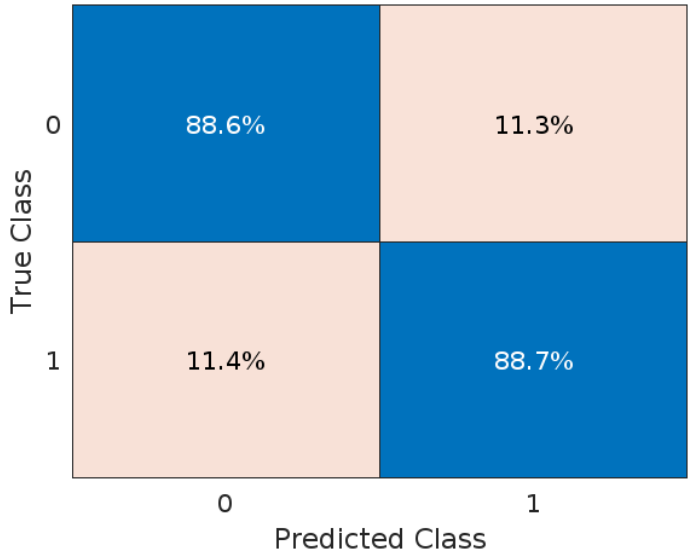

Figura 5.2: Matrice di confusione dell'albero univariato

infatti, viene espressa dalla *mean out-sample accuracy* la quale ne descrive la capacità di generalizzare, ovvero la capacita di prevedere correttamente nuovi dati.

Usando le formule 4.2 del capitolo precedente si sono ottenuti i seguenti risultati:

1. *Mean out-sample accuracy*: **82.0%**.
2. *Richiamo*: **82.5%**.
3. *Precisione*: **85.6%**.

Ovvero nel **82.0%** dei casi l'algoritmo è stato in grado di prevedere la classe corretta di nuovi pazienti, proprietà espressa dalla *mean out-sample accuracy*. Sul totale dei pazienti effettivamente deceduti, il modello è stato in grado di predire correttamente la morte nel **82.5%** dei casi, proprietà espressa dal *richiamo*. Infine nel **85.6%** dei casi in cui l'algoritmo ha predetto un decorso negativo per il paziente quest'ultimo è effettivamente poi deceduto, proprietà espressa dall'indicatore *precisione*.

I risultati ottenuti sono il linea con quanto riportato da *Bertsimas, D., Dunn, J.* [2] dove per il medesimo modello, ad un'altezza uguale a 3, è stata ottenuta un *mean out-sample accuracy* del **79.5%** calcolata testando l'algoritmo su 53 dataset ottenuti da *UCI machine learning repository* [8].

5.1.3 Analisi di sensistività

Per un'analisi più dettagliata del modello è stata svolta un'analisi di sensitività rispetto all'altezza dell'albero, ovvero si è valutata la risposta del modello a seguito del cambiamento del parametro D.

In particolare per $D = 2$ si sono ottenuti i seguenti risultati:

1. *Mean out-sample accuracy*: **68.7%**.
2. *Richiamo*: **70.0%**.
3. *Precisione*: **76.0%**.

Mentre per $D = 4$ si sono ottenuti:

1. *Mean out-sample accuracy*: **73.3%**.
2. *Richiamo*: **76.0%**.
3. *Precisione*: **79.5%**.

Non sono stati tenuti in considerazione valori di D maggiori a 4 in quanto già all'altezza 3 si vede che il modello non necessità di altri nodi, in quanto le foglie non sono pienamente sfruttate.

Questi risultati mostrano come effettivamente il valore $D = 3$ dell'altezza sia quello ottimale per l'albero in questione, infatti quando il parametro viene settato a un valore diverso da questo si ottengono risultati più scadenti.

Quando $D = 2$ si ottengono risultati inferiori in quanto l'albero non è in grado di cogliere pienamente la struttura sottostante ai dati a causa della sua semplicità, infatti si hanno solo 3 *branch node*, ovvero, il numero massimo di variabili sulla quale è possibile effettuare uno split sono 3 e questa può essere una forte limitazione per alcuni *dataset*, come per esempio questo.

Quando $D = 4$ invece, teoricamente si dovrebbero ottenere risultati almeno pari a quelli ottenuti con $D = 3$ in quanto il primo insieme di soluzioni racchiude il secondo. Non è raro però ottenere delle soluzioni con un grado di accuratezza inferiore in quanto all'aumentare del parametro D il numero di equazioni aumenta esponenzialmente (il

numero totale di nodi dell'albero segue la progressione 2^D) il che comporta una maggiore difficoltà per il risolutore nel trovare la soluzione ottima, o comunque una che le si avvicini il più possibile.

5.2 Modello multivariato

5.2.1 Struttura dell'albero

Il modello mutivariato, addestrato su tutti i 150 dati a disposizione, ha prodotto i seguenti risultati:

a [*,*]:	1	2	3	4	5	6	7
covid test	0	0	0	0	0	0	0
age	0	0	0	0	0	0	0
cronic	-0.000799558	0	0	0	0	0	0
diabets	0	0	0	0	0	0	0
respiratory	-0.00174844	0	0	0	0	0	0
kidney	0	0	0	0	0	0	0
neutr	-0.01015	0	0	0	0	0	0
d-dimero	0	0	0	0	0	0	0
AST	0	0	0	0	0	0	0
LDH	0	0	0	0	0	0	0
CK	0	0	0	0	0	0	0
PCR	0	0	0	0	0	0	0
cTnL	-0.657713	0	0	0	0	0	0
WBC/Linf	0	0	0	0	0	0	0
P/F	0.00752623	0	0	0	0	0	0

Tabella 5.5: Soluzione relativa alla variabile $\mathbf{a}_t$.

b[*]:	
1	-0.00138119
2	0
3	0
4	0
5	0
6	0
7	0

Tabella 5.6: Soluzione relativa alla variabile b

L_t[*]:	
8	0
9	0
10	0
11	9
12	0
13	0
14	0
15	8

Tabella 5.7: Soluzione relativa alla variabile L_t

c_kt:		
0	8	0
0	9	0
0	10	0
0	11	0
0	12	0
0	13	0
0	14	0
0	15	1
1	8	0
1	9	0
1	10	0
1	11	1
1	12	0
1	13	0
1	14	0
1	15	0

Tabella 5.8: Soluzione relativa alla variabile c_{kt}

Sfortunatamente in questo caso non è più ammessa una rappresentazione dell'albero come fatto nella sezione precedente in quanto il criterio di suddivisione è cambiato: infatti se nel modello univariato prima si selezionava una *feature* e successivamente veniva scelta la soglia b attraverso la quale la *feature* veniva smistata, adesso la suddivisione viene fatta considerando una combinazione lineare delle stesse.

Per capire meglio il nuovo processo di suddivisione si consideri per semplicità il caso $\mathbf{a}_t$, $\mathbf{x} \in \mathbb{R}^2$: nel modello univariato questo algoritmo va a partizionare ricorsivamente l'insieme $[0, 1]^2$ attraverso rette che possono assumere solo due direzioni, quella parallela all'asse delle x (quando viene selezionata la seconda *feature*) e quella prallela all'asse delle y (quando viene selezionata la prima *feature*), mentre in quello multivariato le rette possono assumere una qualsiasi pendenza.

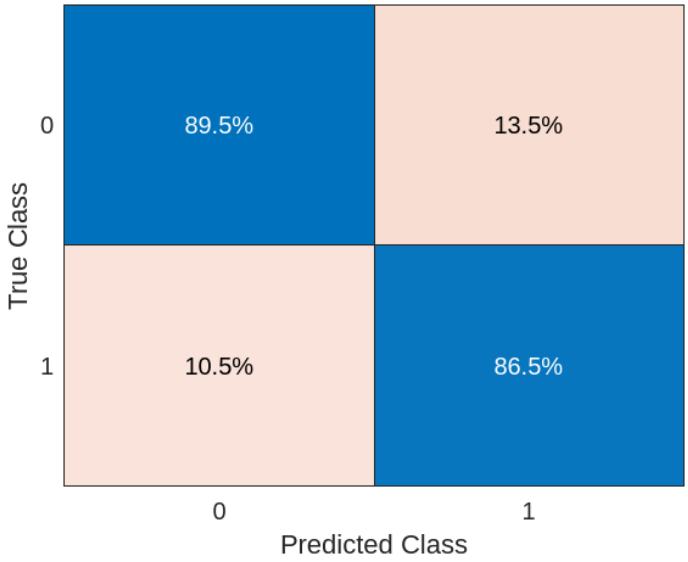

Figura 5.3: Matrice di confusione dell'albero multivariato

E' proprio per la proprietà di aver a disposizione un set più ampio di possibilità di partizionamento di un insieme che, *in media*, il modello multivariato produce risultati migliori dell'univariarto utilizzando anche meno nodi dell'albero.

5.2.2 Proprietà dell'albero

L'albero esposto nella precedente sezione viene caratterizzato dalla *matrice di confusione* della figura 5.3:

Come si può vedere l'algoritmo ha una percentuale dell'**89.5%** nel classificare correttamente i pazienti sopravvissuti, mentre giunge alla conclusione erronea che un paziente sopravviverà solo nel **10.5%** dei casi. Con un ragionamento analogo, si può dire che l'algoritmo ha una percentuale di successo dell'**86.5%** nel classificare in modo idoneo i pazienti che sono morti mentre nel **13.5%** dei casi l'algoritmo è giunto alla conclusione sbagliata della morte di un paziente.

Usando le formule (4.2) del capitolo precedente si sono ottenuti i seguenti risultati:

1. *Mean out-sample accuracy*: **83.5%**.
2. *Richiamo*: **86.0%**.
3. *Precisione*: **88.2%**.

Ovvero nel **83.5%** dei casi l'algoritmo è stato in grado di prevedere la classe corretta di nuovi pazienti, proprietà espressa dalla *mean out-sample accuracy*. Sul totale dei pazienti effettivamente deceduti, il modello è stato in grado di predire correttamente la morte nel **86.0%** dei casi, proprietà espressa dal *richiamo*. Infine nel **88.2%** dei casi in cui l'algoritmo ha predetto un decorso negativo per il paziente quest'ultimo è effettivamente poi deceduto, proprietà espressa dall'indicatore *precisione*.

Anche in questo caso i risultati sono in linea con quanto ottenuto da *Bertsimas, D., Dunn, J.* [2] dove per il medesimo modello, ad un'altezza uguale a 3, è stata ottenuta un *mean out-sample accuracy* del 82.3% calcolata testando l'algoritmo su 53 dataset ottenuti da *UCI machine learning repository* [8].

5.2.3 Analisi di sensitività

L'analisi di sensitività di questo modello è stata condotta testando gli stessi valori del parametro D che si sono testati nel modello univariato.

In particolare per $D = 2$ si sono ottenuti:

1. *Mean out-sample accuracy*: **82.6%**.
2. *Richiamo*: **80.0%**.
3. *Precisione*: **87.5%**.

Mentre per $D = 4$ si sono ottenuti:

1. *Mean out-sample accuracy*: **82.6%**.
2. *Richiamo*: **81.0%**.
3. *Precisione*: **87.7%**.

Come si può notare, già ad un'altezza 2 si possono ottenere risultati molto vicini a quelli ottimali. Questo risultato può essere spiegato dal fatto che nel modello multivariato non si ha più il vincolo di scegliere una sola variabile ad ogni *branch node* bensì, potenzialmente, la scelta potrebbe ricadere su tutte e 15, avendo così a disposizione un maggior numero di possibilità per il partizionamento dell'albero.

Anche se già con $D = 2$ si ottengono risultati molto vicini all'ottimalità si è deciso di usare un'altezza finale di 3 in quanto in futuro potrebbero arrivare nuovi dati e con essi potrebbe cambiare la soluzione del modello di conseguenza si è preferito lasciar un maggior grado di libertà.

Capitolo 6

Un modello di ottimizzazione robusta per l'albero decisionale ottimizzato

In questo capitolo verranno introdotte tre tecniche che permettono di gestire l'incertezza nei dati, infatti, nella vita reale, questi non raramente sono soggetti a rumore il quale ha un influenza negativa sulle prestazioni dell'algoritmo.

In particolare verranno formulati modelli robusti con insiemi di incertezza con la forma di iperrettangoli o iperellissoidi e un modello di ottimizzazione distributivamente robusto basato sui momenti che applica limiti alle deviazioni di primo ordine lungo le direzioni principali.

6.1 Albero di classificazione ottimo e robusto

In questo capitolo si assume che ogni osservazione $\mathbf{x}_i \in \mathbb{R}^p, i = 1,...,N$ sia soggetta ad incertezza, ovvero, ad ognuna di queste osservazioni viene assegnato un insieme di possibili valori, insieme che viene rappresentato da $\mathscr{U}(\mathbf{x}_i)$.

La controparte robusta del modello (2.40) viene definita considerando per ogni osservazione $\mathbf{x}_i$ la sua realizzazione nel caso peggiore, ovvero, il modello può essere riscritto come:

$$\max_{\mathbf{x}\in\mathscr{U}(\mathbf{x}_i)} [\mathbf{a}_m^\top \mathbf{x}] + \mu \leq b_m + M_1(1 - z_{it}), \quad i = 1,\ldots,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t), \tag{6.1}$$

$$\min_{\mathbf{x}\in\mathscr{U}(\mathbf{x}_i)} [\mathbf{a}_m^\top \mathbf{x}] \geq b_m - M_2(1 - z_{it}), \quad i = 1,\ldots,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_R(t), \tag{6.2}$$

dove per semplicità sono state riportate solo le equazioni oggetto di modifica.

A seconda del metodo utilizzato l'insieme $\mathscr{U}(\mathbf{x_i})$ assumerà una forma diversa. Si rimanda alle sezioni successive per una descrizione più dettagliata.

6.2 Albero di classificazione con robustezza ad iper-rettangolo

Con la tecnica chiamata albero di classificazione con robustezza ad iper-rettangolo l'insieme d'incertezza $\mathscr{U}(\mathbf{x}_i)$ assume la forma di un iperrettangolo.

Sia $\xi_{x_i} \in \mathbb{R}^p_+$ il vettore perturbazione dell'osservazione $\mathbf{x}_i$ e $\rho_x \in \mathbb{R}_+$ uno scalare che definisce la misura d'incertezza globale delle osservazioni, ovvero la grandezza degli iperrettangoli. Si definisce l'insieme d'incertezza ad iperrettangolo $\mathscr{U}_B(\mathbf{x}_i)$ centrato in $\mathbf{x}_i$ come:

$$\mathscr{U}_B(\mathbf{x}_i) := \left\{ \mathbf{x} \in \mathbb{R}^p \middle| \mathbf{x}_i - \rho_x \xi_{x_i} \leq \mathbf{x} \leq \mathbf{x}_i + \rho_x \xi_{x_i} \right\} \quad i = 1, \ldots, N. \tag{6.3}$$

Il modello (2.40) viene quindi riscritto come:

$$min \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} \sum_{j=1}^{p} s_{jt} \tag{6.4}$$

t.c.

...

$$\mathbf{a}_m^\top \mathbf{x} + \mu \leq b_m + M_1(1 - z_{it}), \quad \forall \mathbf{x} \in \mathscr{U}_B(\mathbf{x_i}) \quad i = 1, \ldots, N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t),$$

$$\mathbf{a}_m^\top \mathbf{x} \geq b_m - M_2(1 - z_{it}), \quad \forall \mathbf{x} \in \mathscr{U}_B(\mathbf{x_i}) \quad i = 1, \ldots, N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_R(t),$$

...

il quale può essere rappresentato nel seguente modo:

$$min \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} \sum_{j=1}^{p} s_{jt} \tag{6.5}$$

t.c.

...

$$\max_{\mathbf{x} \in \mathscr{U}_B(\mathbf{x}_i)} \left[\mathbf{a}_m^\top \mathbf{x} \right] + \mu \leq b_m + M_1(1 - z_{it}), \quad i = 1, \ldots, N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t),$$

$$\min_{\mathbf{x} \in \mathscr{U}_B(\mathbf{x}_i)} \left[\mathbf{a}_m^\top \mathbf{x} \right] \geq b_m - M_2(1 - z_{it}), \quad i = 1, \ldots, N, \quad \forall t \in \{T_L\}. \quad \forall m \in A_R(t),$$

...

La parte sinistra del primo vincolo in (6.5) può essere riscritta come:

$$\max_{\mathbf{x}} \ \mathbf{a}_m^\top \mathbf{x} \tag{6.6}$$

t.c.

$$\mathbf{x} \leq \mathbf{x_i} + \rho_x \xi_{x_i}$$

$$\mathbf{x} \geq \mathbf{x}_i - \rho_x \xi_{x_i},$$

la quale ammette il seguente problema duale:

$$\min_{\mathbf{a}^+, \mathbf{a}^-} \ (\mathbf{x}_i + \rho_x \xi_{x_i})^\top \mathbf{a}^+ - (\mathbf{x}_i - \rho_x \xi_{x_i})^\top \mathbf{a}^- \tag{6.7}$$

t.c.

$$\mathbf{a}^+ - \mathbf{a}^- = \mathbf{a}$$

$$\mathbf{a}^+ \geq 0, \quad \mathbf{a}^- \geq 0,$$

il quale può essere equivalentemente riscritto:

$$\min_{\mathbf{a}^+ - \mathbf{a}^-} \ (\mathbf{a}^+ - \mathbf{a}^-)^\top \mathbf{x}_i + \rho_x \xi_{x_i}^\top (\mathbf{a}^+ - \mathbf{a}^-)^\top \tag{6.8}$$

t.c.

$$\mathbf{a}^+ - \mathbf{a}^- = \mathbf{a}$$

$$\mathbf{a}^+ \geq 0, \quad \mathbf{a}^- \geq 0,$$

che corrisponde a:

$$\min_{\mathbf{a}} \ \mathbf{a}^\top \mathbf{x}_i + \rho_x \xi_{x_i}^\top |\mathbf{a}|. \tag{6.9}$$

Il secondo vincolo in (6.5) ammette la seguente equivalenza:

$$\min_{\mathbf{x} \in \mathscr{U}_B(\mathbf{x}_i)} \left[\mathbf{a}_m^\top \mathbf{x} \right] \geq b_m - M_2(1 - z_{it}) \longleftrightarrow \max_{\mathbf{x} \in \mathscr{U}_B(\mathbf{x}_i)} \left[-\mathbf{a}_m^\top \mathbf{x} \right] \leq -b_m + M_2(1 - z_{it})$$

Di conseguenza si giunge alla medesima conclusione anche per il secondo vincolo. Infine il modello (6.4) può essere riscritto come:

$$min \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} \sum_{j=1}^{p} s_{jt} \tag{6.10}$$

t.c.

...

$$\mathbf{a}^\top \mathbf{x}_i + \rho_x \xi_{x_i}^\top |\mathbf{a}| + \mu \leq b_m + M_1(1 - z_{it}), \quad i = 1, \ldots, N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t),$$

$$\mathbf{a}^\top \mathbf{x}_i - \rho_x \xi_{x_i}^\top |\mathbf{a}| \geq b_m - M_2(1 - z_{it}), \quad i = 1, \ldots, N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_R(t),$$

...

Come si può notare l'unica modifica al modello riguarda le equazioni (2.34) e (2.35) che vedono l'aggiunta di due ulteriori variabili: ξ_{x_i} e ρ_x.

In particolare il vettore ξ_{x_i} è da interpretare come la deviazione standard delle osservazioni mentre il parametro $\rho_x \in \{0.1, 0.01, 0.001\}$ è stato testato per ciascuno di questi valori al fine di valutare la risposta del modello al variare della grandezza degli iperrettangoli.

6.2.1 Implementazione del modello in AMPL

La modifica al modello scritto in AMPL è stata la seguente:

```
subject to eq8_1 {i in OSSERVAZIONI, t in LEAF_NODES, a_l
   in A_L[t]}:
sum {j in FEATURES} (a[j,a_l]*x[i,j]) + ro*(sum {j in
   FEATURES} a_capp[j,a_l]*( std_x_classe1[j]*(y[i]) +
   std_x_classe0[j]*(1 - y[i]) )) + mu <= b[a_l] + (3 +
   mu)*(1 - z[i,t]);

subject to eq8_2 {i in OSSERVAZIONI, t in LEAF_NODES, a_r
   in A_R[t]}:
sum {j in FEATURES} (a[j,a_r]*x[i,j]) - ro*(sum {j in
   FEATURES} a_capp[j,a_r]*( std_x_classe1[j]*(y[i]) +
   std_x_classe0[j]*(1 - y[i]) )) >= b[a_r] - 3*(1 -
   z[i,t]);
```

Si noti che non è stato necessario linearizzare il valore assoluto dei vincoli in (6.10) in quanto basta usare la variabile *a_capp* calcolata precedentemente. Inoltre per la deviazione standard sono stati usati due vettori differenti per classe, questo al fine di catturare al meglio la struttura sottostante dei dati. Infine le costanti M_1 e M_2 sono state modificate attraverso l'assegnamento del valore 3: questa modifica è stata effettuata per tenere conto del termine additivo che è stato introdotto nel nuovo modello.

6.2.2 Risultati numerici

Al fine di testare la bontà del modello sono stati utilizzati gli indicatori (4.1). E' stato quindi eseguito il metodo della *K-fold cross validation* per ognuno dei tre valori del parametro ρ ottenendo i seguenti risultati.

Caso $\rho = 0.1$:

1. *Mean out-sample accuracy*: **58.0%**.
2. *Richiamo*: **50.0%**.
3. *Precisione*:0 **77.0%**.

Caso $\rho = 0.01$:

1. *Mean out-sample accuracy*: **79.3%**.
2. *Richiamo*: **79.6%**.
3. *Precisione*: **83.5%**.

Caso $\rho = 0.001$:

1. *Mean out-sample accuracy*: **79.33%**.
2. *Richiamo*: **78.0%**.
3. *Precisione*: **82.6%**.

Caso $\rho = 0.0001$:

1. *Mean out-sample accuracy*: **76.0%**.
2. *Richiamo*: **70.7%**.
3. *Precisione*: **81.7%**.

Caso $\rho = 0.00001$:

1. *Mean out-sample accuracy*: **76.6%**.
2. *Richiamo*: **78.6%**.
3. *Precisione*: **83.6%**.

6.3 Albero di classificazione con robustezza ellissoidale

Il problema del metodo precedentemente esposto è che può portare a soluzioni eccessivamente conservative, di conseguenza, per alleviare a questo inconveniente in questa sezione viene proposta una riformulazione alternativa del modello (6.4) che considera insiemi di incertezza aventi la forma di iperellissoidi.

Questa scelta permette di ottenere modelli meno conservativi rispetto agli iperrettangoli in quanto vengono ignorate quelle situazioni in cui le *features* assumono congiuntamente valori di intervallo estremi.

Si consideri il probelma:

$$min \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} \sum_{j=1}^{p} s_{jt} \tag{6.11}$$

t.c.

...

$$\mathbf{a}_m^\top \mathbf{x} + \mu \leq b_m + M_1(1 - z_{it}), \quad \forall \mathbf{x} \in \mathscr{U}(\mathbf{x_i}) \quad i = 1, \ldots, N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t),$$

$$\mathbf{a}_m^\top \mathbf{x} \geq b_m - M_2(1 - z_{it}), \quad \forall \mathbf{x} \in \mathscr{U}(\mathbf{x_i}) \quad i = 1, \ldots, N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_R(t),$$

...

dove:

$$\mathscr{U}(\mathbf{x_i}) := \left\{ \mathbf{x} \in \mathbb{R}^p \;\middle|\; \begin{array}{c} \mathbf{x} = \mathbf{x_i} + \Sigma_{x_i}^{1/2} \mathbf{u} \\ \|\mathbf{u}\|_2 \leq \rho_x \end{array} \right\}, \tag{6.12}$$

rappresenta l'insieme d'incertezza dell'osservazione $\mathbf{x_i}$ avente la forma di iperellissoide.

$\Sigma_{x_i} \in \mathbb{R}^{p*p}$ è la matrice di covarianza definita positiva associata all'osservazione $\mathbf{x_i}$ e $\rho_x \in \mathbb{R}_+$ il raggio degli ellissoidi delle stesse osservazioni.

L'insieme (6.12) può essere riscritto nella seguente forma:

$$\mathscr{U}(\mathbf{x_i}) := \left\{ \mathbf{x} \in \mathbb{R}^p | (\mathbf{x} - \mathbf{x_i})^\top \Sigma_{x_i}^{-1} (\mathbf{x} - \mathbf{x_i}) \leq \rho_x^2 \right\} \quad i = 1, .., N. \tag{6.13}$$

Come fatto per il metodo *Box robust* il problema (6.11) può essere riscritto nella seguente forma:

$$min \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} \sum_{j=1}^{p} s_{jt} \tag{6.14}$$

t.c.

...

$$\max_{\mathbf{x} \in \mathscr{U}(\mathbf{x_i})} \left[\mathbf{a}_m^\top \mathbf{x} \right] + \mu \leq b_m + M_1(1 - z_{it}), \quad i = 1, \ldots, N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t),$$

$$\min_{\mathbf{x} \in \mathscr{U}(\mathbf{x_i})} \left[\mathbf{a}_m^\top \mathbf{x} \right] \geq b_m - M_2(1 - z_{it}), \quad i = 1, \ldots, N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_R(t),$$

...

La parte sinistra del primo vincolo può essere riscritta come:

$$\max_{\mathbf{x}} \ \mathbf{a}_m^\top \mathbf{x} \tag{6.15}$$

t.c.

$$\mathbf{x} \in \mathscr{U}(\mathbf{x_i}),$$

che è equivalente a:

$$\mathbf{a}_m^\top \mathbf{x}_i + \max_{u} \left[\mathbf{a}_m^\top \Sigma_{x_i}^{1/2} \mathbf{u} \right] \tag{6.16}$$

t.c.

$$\|\mathbf{u}\|_2 \leq \rho_x.$$

Applicando la diseguaglianza di *Cauchy-Schwarz* si ottiene:

$$|\mathbf{a}_m^\top \Sigma_{x_i}^{1/2} \mathbf{u}| \leq \|\mathbf{a}_m^\top \Sigma_{x_i}^{1/2}\|_2 * \|\mathbf{u}\|_2,$$

ed essendo $\|\mathbf{u}\|_2 \leq \rho_x$ si ottiene:

$$\mathbf{a}_m^\top \mathbf{x}_i + \|\mathbf{a}_m^\top \Sigma_{x_i}^{1/2}\|_2 * \|\mathbf{u}\|_2 \longleftrightarrow \mathbf{a}_m^\top \mathbf{x}_i + \rho_x \|\mathbf{a}_m^\top \Sigma_{x_i}^{1/2}\|_2. \tag{6.17}$$

La medesima procedura può essere eseguita per il secondo vincolo ricordando che:

$$\min_{\mathbf{x}\in\mathscr{U}(\mathbf{x_i})} \left[\mathbf{a}_m^\top \mathbf{x}\right] \geq b_m - M_2(1-z_{it}) \longleftrightarrow \max_{\mathbf{x}\in\mathscr{U}(\mathbf{x_i})} \left[-\mathbf{a}_m^\top \mathbf{x}\right] \leq -b_m + M_2(1-z_{it}).$$

In conclusione la controparte robusta del modello (6.11) può essere riformulata come:

$$min \frac{1}{\hat{L}} \sum_{t\in T_L} L_t + \alpha \sum_{t\in T_B} \sum_{j=1}^{p} s_{jt} \tag{6.18}$$

t.c.

...

$$\mathbf{a}_m^\top \mathbf{x}_i + \rho_x \|\mathbf{a}_m^\top \Sigma_{x_i}^{1/2}\|_2 + \mu \leq b_m + M_1(1-z_{it}), \quad i = 1,\dots,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t),$$

$$\mathbf{a}_m^\top \mathbf{x}_i - \rho_x \|\mathbf{a}_m^\top \Sigma_{x_i}^{1/2}\|_2 \geq b_m - M_2(1-z_{it}), \quad i = 1,\dots,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_R(t),$$

...

Dove $\|\mathbf{a}_m^\top \Sigma_{x_i}^{1/2}\|_2 := \sqrt{\mathbf{a}_m^T \Sigma_{x_i} \mathbf{a}_m}$.

Anche in questo caso l'unica modifica al modello riguarda le equazioni (2.34) e (2.35) che vedono l'aggiunta di due ulteriori variabili: Σ_{x_i} e ρ_x.

In pratica la matrice $\Sigma_{x_i}^{1/2}$ è la stessa per ogni osservazione ed è costruita nel seguente modo:

$$\Sigma_{x_i}^{1/2} = diag(\xi_{\mathbf{x_i}}) \quad \forall i = 1,...,N,$$

dove ξ_{x_i} è il vettore delle deviazioni standard introdotto in (6.10). Come si evince $\Sigma_{x_i}^{1/2}$ è una matrice diagonale con elementi le deviazioni standard lungo le componenti delle osservazioni.

Per il parametro ρ_x, ovvero il raggio degli ellissoidi, sono stati presi in considerazione diversi valori: 0.1, 0.01, 0.001, 0.0001, 0.0000 al fine di valutare la risposta del modello al variare della grandezza degli iperellissoidi.

6.3.1 Implementazione del modello in AMPL

La modifica al modello scritto in AMPL è stata la seguente:

```
subject to eq8_1 {i in OSSERVAZIONI, t in LEAF_NODES, a_l
   in A_L[t]}:
sum {j in FEATURES} ( a[j,a_l]*x[i,j] ) + ro*sqrt( sum {j
   in FEATURES} a[j,a_l]*( cov_matrix_classe1[j,j]*y[i] +
   cov_matrix_classe0[j,j]*(1-y[i]) )*a[j,a_l] ) + mu <=
   b[a_l] + (3 + mu)*(1 - z[i,t]);

subject to eq8_2 {i in OSSERVAZIONI, t in LEAF_NODES, a_r
   in A_R[t]}:
sum {j in FEATURES} (a[j,a_r]*x[i,j]) - ro*sqrt( sum {j in
   FEATURES} a[j,a_r]*( cov_matrix_classe1[j,j]*y[i] +
   cov_matrix_classe0[j,j]*(1-y[i]) )*a[j,a_r] ) >= b[a_r]
   - 3*(1 - z[i,t]);
```

Come nel modello ad iperrettangolo si è scelto di usare due valori diversi per la matrice delle covarianze Σ_{x_i} identificati rispettivamente dai parametri *cov_matrix_classe1* e *cov_matrix_classe0* del modello al fine di tenere in considerazione le diverse varianze delle due classi.

6.3.2 Risultati numerici

Al fine di testare la bontà del modello sono stati utilizzati gli indicatori (4.1). E' stato quindi eseguito il metodo della *K-fold cross validation* per ognuno dei valori del parametro ρ ottenendo i seguenti risultati:

Caso $\rho = 0.1$:

1. *Mean out-sample accuracy*: **79.3%**.
2. *Richiamo*: **69.69%**.
3. *Precisione*: **71.32%**.

Caso $\rho = 0.01$:

1. *Mean out-sample accuracy*: **76.6%**.
2. *Richiamo*: **76.6%**.
3. *Precisione*: **83.77%**.

Caso $\rho = 0.001$:

1. *Mean out-sample accuracy*: **76.0%**.
2. *Richiamo*: **76.93%**.
3. *Precisione*: **82.2%**.

Caso $\rho = 0.0001$:

1. *Mean out-sample accuracy*: **74.6%**.
2. *Richiamo*: **73.10%**.
3. *Precisione*: **82.32%**.

Caso $\rho = 0.00001$:

1. *Mean out-sample accuracy*: **82.66%**.
2. *Richiamo*: **84.86%**.
3. *Precisione*: **88.33%**.

Caso $\rho = 0.00001$, $\mu = 0.000005$:

1. *Mean out-sample accuracy*: **85.33%**.
2. *Richiamo*: **86.10%**.
3. *Precisione*: **92.33%**.

Si noti che negli ultimi due casi il valore di ρ è il medesimo ma si sono ottenuti due risultati differenti, questo viene spiegato dal fatto che quando il raggio diventa molto piccolo il parametro μ scelto precedentemente interferisce con la soluzione, di conseguenza è necessario diminuirlo e impostare i parametri del solutore *IntFeasTol* e *FeasTol* entrambi uguali a $1e^{-09}$ che è il valore più piccolo permesso.

6.4 Albero di classificazione distribuzionalmente robusto

Le soluzioni ottenute considerando insiemi d'incertezza aventi la forma di iperellissoidi possono essere ancora conservative. Un modo per superare questa limitazione è considerare altre forme d'incertezza come: poliedriche, coniche, vincoli convessi (Gorissen et al. 2015 [7]) oppure una combinazione delle stesse con la limitazione di richiedere una specifica conoscenza delle istanze sotto analisi, spesso non accessibile. Un metodo per superare queste limitazioni, ottenendo soluzioni meno conservative e allo stesso tempo senza perdere la proprietà di generalizzazione sono le tecniche basate sull'ottimizzazione distribuzionalmente robusta DRO (Distributionally Robust Optimization) [10].

In questa sezione si assume che le osservazioni $\mathbf{x}_i,\ i = 1,\dots,N$ siano variabili casuali per le quali le esatte distribuzioni di probabilità $\mathbb{P}^{true}_{\mathbf{x}_i}$ si suppongono sconosciute.

Per proteggersi dall'incertezza, per ogni osservazione in input si ottimizza contro l'aspettativa nel caso peggiore considerando tutte le distribuzioni $\mathbb{P}$ appartenenti all'insieme di ambiguità $\mathscr{D}(\mathbf{x}_i)$, ovvero il modello (2.40) può essere riscritto come:

$$
\begin{aligned}
min \quad & \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} \sum_{j=1}^{p} s_{jt} && (6.19)\\
s.t. \quad & \\
& \dots \\
& \sup_{\mathbb{P} \in \mathscr{D}(\mathbf{x}_i)} \mathbb{E}_{\mathbb{P}} \left[\mathbf{a}_m^\top \mathbf{x} \right] + \mu \leq b_m + M_1(1 - z_{it}), \quad i = 1,\dots,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t), \\
& \inf_{\mathbb{P} \in \mathscr{D}(\mathbf{x}_i)} \mathbb{E}_{\mathbb{P}} \left[\mathbf{a}_m^\top \mathbf{x} \right] \geq b_m - M_2(1 - z_{it}), \quad i = 1,\dots,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_R(t), \\
& \dots
\end{aligned}
$$

dove:

$$\mathscr{D}(\mathbf{x}_i) := \left\{ \mathbb{P} \in \mathscr{P}_+^p \;\middle|\; \begin{array}{c} \mathbb{P}\Big(\mathbf{x} \in \mathscr{U}_B(\mathbf{x}_i)\Big) = 1 \\ \mathbb{E}_{\mathbb{P}}\Big[g_p(\mathbf{x})\Big] \leq (\vartheta_x)_p \quad p = 1,...,P, \end{array} \right\} \quad i = 1,...,N. \tag{6.20}$$

Con $\mathscr{P}_+^p$ si indica l'insieme contenente le distribuzioni di probabilità su $\mathbb{R}_+^p$, $\mathscr{U}_B(\mathbf{x}_i)$ insieme di supporto a forma di iperrettangolo definito in (6.4). Il secondo insieme di vincoli caratterizzano l'informazione dei momenti attraverso p funzioni $g_p(\cdot)$ e constringe i momenti generalizzati a non superare un limite $(\vartheta_x)_p \in \mathbb{R}^+ \;\; p = 1,...,P$. Le funzioni $g_p(\mathbf{x}) \;\; p = 1,\ldots,P$ possono essere modellizzate per semplicità come le deviazioni del primo ordine del parametro incerto $\mathbf{x}$ rispetto al valore nominale $\mathbf{x}_i$ lungo le componenti principali, ovvero lungo le direzioni degli autovettori della matrice di covarianza Σ_x.

Più sinteticamente:

$$g_p(\mathbf{x}) := \left| {\mathbf{f}_x^{(p)}}^\top (\mathbf{x} - \mathbf{x}_i) \right| \quad p = 1,\ldots,P. \tag{6.21}$$

Per determinare il vettore delle componenti principali $\mathbf{F}_x := \left[\mathbf{f}_x^{(1)},...,\mathbf{f}_x^{(N)}\right] \in \mathbb{R}^{p*p}$ si adotta la tecnica di Analisi delle Componenti Principali (PCA).

La PCA fa parte delle tecniche statistiche di apprendimento non supervisionato ovvero quelle tecniche che esplorano i dati non alla ricerca di un qualcosa di definito a priori bensì dalla ricerca di proprietà intrinseche o nascoste nei dati.

L'obiettivo principale è quello di ridurre il numero di variabili iniziali perdendo il minor numero di informazioni possibile. Dal punto di vista matematico significa trovare una combinazione lineare delle variabili di partenza che permettono di ridurre la dimensione del *dataset* iniziale e al tempo stesso mantenere una quota medio-alta di variabilità spiegata.

Il risultato di tale processo dal punto di vista geometrico è un nuovo spazio, di dimensione ridotta rispetto a quella di partenza, in cui verrà proiettata la nube di punti iniziale e dove le distanze originarie tra i punti stessi saranno deformate il meno possibile. Dal punto di vista quantitativo invece permette di ottenere le componenti principali, ovvero le direzioni che contengono la maggiore varianza dei dati, ordinate in senso decrescente di varianza e incorrelate a due a due, cioè $Cov(CP_i, CP_j) = 0, \;\; i \neq j$.

In virtù di ciò il primo passo è stimare la matrice di covarianza attraverso:

$$\Sigma_x := \frac{(\sum_{i=1}^{N} \mathbf{x}_i^\top \mathbf{x}_i) - (\sum_{i=1}^{N} \mathbf{x}_i)^\top (\sum_{i=1}^{N} \mathbf{x}_i)}{N-1}. \tag{6.22}$$

Successivamente si calcola il vettore delle componenti principali attraverso la relazione:

$$\Sigma_x = \mathbf{F}_x \Lambda_x \mathbf{F}_x^\top, \tag{6.23}$$

dove $\Lambda_x := diag(\lambda_x) \in \mathbb{R}_+^{p*p}$ rappresenta la matrice diagonale contenente gli autovalori della matrice Σ, ovvero le varianze delle componenti.

Per determinare la massima deviazione ammessa lungo le p direzioni principali si definisce la soglia ϑ_x nel seguente modo:

$$(\vartheta_x)_p := \frac{\rho_x \sqrt{(\lambda_x)_p}}{K} \qquad p = 1, \ldots, P, \tag{6.24}$$

dove $K \in \mathbb{N} \setminus \{0\}$ è un parametro di scala che permette di controllare il grado di conservatorismo del modello, in particolare più K è piccolo meno il modello è conservativo, al contrario più K è grande più il modello è conservativo.

Un esempio di quanto detto è mostrato in figura 6.1 dove viene mostrata la tecnica PCA con 2 valori diversi di K.

A causa del numero infinito di distribuzioni di probabilità contenute nell'insieme d'incertezza (6.20) il modello (6.19) è intrattabile così com'è stato definito, di conseguenza bisogna proporre una formulazione alternativa dello stesso.

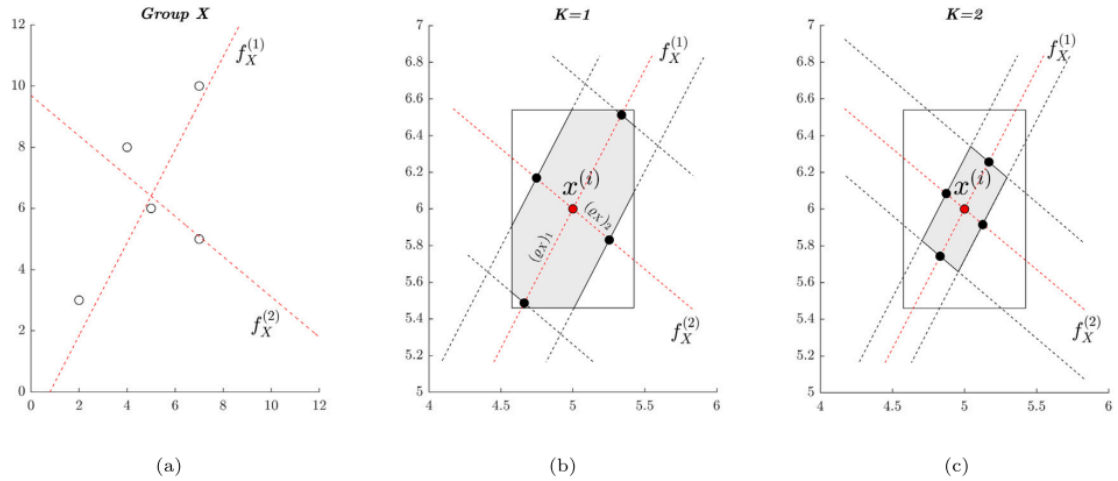

Figura 6.1: La prima immagine mostra, per un gruppo di osservazioni X, le componenti principali $\mathbf{f}_x^{(1)}, \mathbf{f}_x^{(2)}$ calcolate attraverso la PCA, (si noti il fatto che sono in ordine decrescente di varianza). Per ogni osservazione $\mathbf{x}_i$, il suo insieme di supporto $\mathscr{U}_B(\mathbf{x}_i)$ viene rappresentato dal rettangolo verticale contenente l'osservazione. Infine le figure (b) e (c) mostrano come la soluzione sia più o meno conservativa al variare del parametro K. Per $K = 1$ sia ha una soluzione meno conservativa rispetto a $K = 2$.

Fonte: Computers and Operations Research, Robust and Distributionally Robust Optimization Models for Linear Support Vector Machine [4].

Sia $\varphi_x := \Big[(\varphi_x)_1, ..., (\varphi_x)_p\Big] \in \mathbb{R}^{\mathbb{p}}_+$ un vettore ausiliario di valori random, l'insieme (6.20) può essere riscritto come:

$$\bar{\mathscr{D}}(\mathbf{x}_i) := \left\{ \mathbb{Q} \in \mathscr{P}^p_+ \middle| \begin{array}{cc} \mathbb{Q}\Big(\mathbf{x}, \varphi_x \in \bar{\mathscr{U}}_B(\mathbf{x}_i)\Big) = 1 \\ \mathbb{E}_{\mathbb{Q}}\Big[(\varphi_x)_p\Big] \leq (\vartheta_x)_p \quad p = 1, ..., P \end{array} \right\} \quad i = 1, ..., N, \tag{6.25}$$

con l'insieme di supporto modificato:

$$\bar{\mathscr{U}}_B(\mathbf{x}_i) := \left\{ \Big(\mathbf{x}, \varphi_x\Big) \in \mathbb{R}^p \times \mathbb{R}^p_+ \middle| \begin{array}{c} \mathbf{x} \in \mathscr{U}_B(\mathbf{x}_i) \\ g_p(\mathbf{x}) \leq (\varphi_x)_p \quad p = 1, \ldots, P \end{array} \right\}, \tag{6.26}$$

il quale usando (6.3) e (6.21) si può riscrivere come:

$$\bar{\mathscr{U}}_B(\mathbf{x}_i) := \left\{ \Big(\mathbf{x}, \varphi_x\Big) \middle| \begin{array}{c} \mathbf{x} \leq \mathbf{x}_i + \rho_x \xi_{x_i} \\ \mathbf{x} \geq \mathbf{x}_i - \rho_x \xi_{x_i} \\ (\varphi_x)_p \geq 0 \;\; p = 1, \ldots, P \\ {\mathbf{f}_x^{(p)}}^\top \mathbf{x} - {\mathbf{f}_x^{(p)}}^\top \mathbf{x}_i \leq (\varphi_x)_p \;\; p = 1, \ldots, P \\ {\mathbf{f}_x^{(p)}}^\top \mathbf{x}_i - {\mathbf{f}_x^{(p)}}^\top \mathbf{x} \leq (\varphi_x)_p \;\; p = 1, \ldots, P, \end{array} \right\}, \tag{6.27}$$

o equivalentemente nella forma matriciale:

$$\bar{\mathscr{U}}_B(\mathbf{x}_i) := \left\{ \Big(\mathbf{x}, \varphi_x\Big) \middle| \mathbf{C}_x \mathbf{x} + \mathbf{D}\varphi_x \leq \mathbf{h}_i \right\} \quad i = 1, \ldots, P, \tag{6.28}$$

dove:

$$\mathbf{C}_x := \begin{bmatrix} -\mathbf{I} \\ -\mathbf{I} \\ \mathbf{0} \\ \mathbf{F}_x^\top \\ -\mathbf{F}_x^\top \end{bmatrix} \in \mathbb{R}^{5n \times n}, \quad \mathbf{D} := \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{I} \\ -\mathbf{I} \\ -\mathbf{I} \end{bmatrix} \in \mathbb{R}^{5n \times n}, \quad \mathbf{h}_{x_i} := \begin{bmatrix} \mathbf{x}_i + \rho_x \xi_{x_i} \\ -\mathbf{x}_i + \rho_x \xi_{x_i} \\ \mathbf{0} \\ \mathbf{F}_x^\top \mathbf{x}_i \\ -\mathbf{F}_x^\top \mathbf{x}_i \end{bmatrix} \in \mathbb{R}^{5n}. \tag{6.29}$$

Quindi il modello (6.19) può essere riscritto nella seguente forma:

$$min \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} \sum_{j=1}^{p} s_{jt} \tag{6.30}$$

s.t.

$$\ldots$$

$$\sup_{\mathbb{Q} \in \bar{\mathscr{D}}(\mathbf{x}_i)} \mathbb{E}_{\mathbb{Q}}\Big[\mathbf{a}_m^\top \mathbf{x}\Big] + \mu \leq b_m + M_1(1 - z_{it}), \quad i = 1, \ldots, N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t),$$

$$\inf_{\mathbb{Q} \in \bar{\mathscr{D}}(\mathbf{x}_i)} \mathbb{E}_{\mathbb{Q}}\Big[\mathbf{a}_m^\top \mathbf{x}\Big] \geq b_m - M_2(1 - z_{it}), \quad i = 1, \ldots, N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_R(t),$$

$$\ldots$$

Per ogni $i = 1, \ldots, N$ la parte sinistra del primo vincolo di (6.30) coincide con il valore ottimo del seguente problema:

$$\begin{array}{rll} \sup_{\mathbb{Q} \in \bar{\mathscr{D}}(\mathbf{x}_i)} \mathbb{E}_{\mathbb{Q}}\Big[\mathbf{a}_m^\top \mathbf{x}\Big] = & \sup\limits_{\mathbb{Q}} \int_{\bar{\mathscr{U}}_B(\mathbf{x}_i)} q(\mathbf{x}, \varphi_x)(\mathbf{a}_m^\top \mathbf{x}) dx d\varphi_x & \\ s.t. & \int_{\bar{\mathscr{U}}_B(\mathbf{x}_i)} q(\mathbf{x}, \varphi_x) dx d\varphi_x = 1, & (6.31) \end{array}$$

Se esiste una coppia $(\mathbf{x}, \varphi_x)$ tale che $(\mathbf{a}_m^\top \mathbf{x} - \beta_{im}^\top \varphi_x) \geq \eta_{im}$ allora la soluzione di (6.32) è illimitata in quanto $q(\mathbf{x}, \varphi_x) \geq 0, \forall (\mathbf{x}, \varphi_x) \in \bar{\mathscr{U}}_B(\mathbf{x}_i)$. Al contrario se $(\mathbf{a}_m^\top \mathbf{x} - \beta_{im}^\top \varphi_x) \leq \eta_{im} \ \forall (\mathbf{x}, \varphi_x) \in \bar{\mathscr{U}}_B(\mathbf{x}_i)$ il problema (6.32) ammette come soluzione: $\eta_{im} + \beta_{im}^\top \vartheta_x$.

Il duale di (6.31) può essere espresso come:

$$\begin{aligned} \min_{\eta_{im},\beta_{im}} \quad & \eta_{im} + \beta_{im}^{\top}\vartheta_x \\ s.t. \quad & \mathbf{a}_m^{\top}\mathbf{x} - \beta_{im}^{\top}\varphi_x \leq \eta_{im} \quad \forall(\mathbf{x},\varphi_x) \in \bar{\mathscr{U}}_B(\mathbf{x}_i), \\ & \beta_{im} \geq 0, \end{aligned} \tag{6.33}$$

dove per il primo vincolo sussiste la seguente relazione:

$$\mathbf{a}_m^{\top}\mathbf{x} - \beta_{im}^{\top}\varphi_x \leq \eta_{im} \quad \forall(\mathbf{x},\varphi_x) \in \bar{\mathscr{U}}_B(\mathbf{x}_i) \Longleftrightarrow \max_{(\mathbf{x},\varphi_x)\in\bar{\mathscr{U}}_B(\mathbf{x}_i)} \left[\mathbf{a}_m^{\top}\mathbf{x} - \beta_{im}^{\top}\varphi_x\right] \leq \eta_{im}. \tag{6.34}$$

Il problema duale del membro di destra della relazione (6.34) si può esprimere nella forma:

$$\begin{aligned} \min_{\pi_{im}} \quad & \pi_{im}^{\top}\mathbf{h}_i \\ s.t. \quad & \mathbf{C}_x^{\top}\pi_{im} \geq \mathbf{a}_m, \\ & \mathbf{D}^{\top}\pi_{im} \geq -\beta_{im}, \\ & \pi_{im} \geq 0, \end{aligned} \tag{6.35}$$

dove $\pi_{im} \in \mathbb{R}_+^{5n}$. In conclusione, combinando (6.33) e (6.35) si ottiene la formulazione distribuzionalmente robusta del problema (2.40) come:

$$\begin{aligned} min \quad & \frac{1}{\hat{L}}\sum_{t\in T_L} L_t + \alpha \sum_{t\in T_B}\sum_{j=1}^{p} s_{jt} \\ & \ldots \\ & \eta_{im} + \beta_{im}^{\top}\vartheta_x + \mu \leq b_m + M_1(1-z_{it}), \quad i = 1,..,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t), \\ & \pi_{im}^{\top}\mathbf{h}_i \leq \eta_{im}, \quad i = 1,\ldots,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t), \\ & \mathbf{C}_x^{\top}\pi_{im} \geq \mathbf{a}_m \quad i = 1,\ldots,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t), \\ & \mathbf{D}^{\top}\pi_{im} \geq -\beta_{im} \quad i = 1,\ldots,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t), \\ & \ldots \end{aligned} \tag{6.36}$$

Un procedimento analogo può essere applicato al secondo vincolo di (6.30), dove vale la seguente relazione:

$$\inf_{\mathbb{Q}\in\bar{\mathscr{D}}(\mathbf{x}_i)} \mathbb{E}_{\mathbb{Q}}\left[\mathbf{a}_m^T\mathbf{x}\right] \geq b_m - M_2(1-z_{it}) \Longleftrightarrow \sup_{\mathbb{Q}\in\bar{\mathscr{D}}(\mathbf{x}_i)} \mathbb{E}_{\mathbb{Q}}\left[-\mathbf{a}_m^T\mathbf{x}\right] \leq -b_m + M_2(1-z_{it}),$$

potendo così definire le seguenti equazioni:

$$\begin{aligned} min \quad & \frac{1}{\hat{L}}\sum_{t\in T_L} L_t + \alpha \sum_{t\in T_B}\sum_{j=1}^{p} s_{jt} \\ & \ldots \\ & \eta_{im} + \beta_{im}^{\top}\vartheta_x + \mu \leq b_m + M_1(1-z_{it}), \quad i = 1,\ldots,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t), \\ & \pi_{im}^{\top}\mathbf{h}_i \leq \eta_{im} + M_2(1-z_{it}), \quad i = 1,\ldots,N, \quad \forall t \in \{T_L\}, \quad \forall m \in A_L(t), \end{aligned} \tag{6.37}$$

```
subject to eq8_1 {i in OSSERVAZIONI, t in LEAF_NODES, a_l
   in A_L[t]}:
  if z[i,t] == 1 then sum {j in FEATURES}
     (a[j,a_l]*x[i,j]) + mu <= b[a_l];

subject to eq8_3 {i in OSSERVAZIONI, t in LEAF_NODES, a_l
   in A_L[t]}:
  z[i,t] == 1 ==> eta[i,a_l] + sum{jj in FEATURES}
     (beta[i,jj,a_l]*( y[i]*varrho_classe1[jj] +
     (1-y[i])*varrho_classe0[jj] ) ) + mu <= b[a_l];

subject to eq8_3_1_DRO {i in OSSERVAZIONI, t in
   LEAF_NODES, tt in A_L[t]}:
  z[i,t] == 1 ==> sum{k in 1..5}( sum{jj in FEATURES}
     h_i[i,jj,k]*pi_greco[i,jj,tt,k] ) <= eta[i,tt];

subject to eq8_3_2_DRO {i in OSSERVAZIONI, j in FEATURES,
   t in LEAF_NODES, tt in A_L[t]}:
  z[i,t] == 1 ==> sum{k in 1..5}( sum{jj in FEATURES}
     (y[i]*C_x_classe1[j,jj,k] +
     (1-y[i])*C_x_classe0[j,jj,k] )*pi_greco[i,jj,tt,k] )
     >= a[j,tt];

subject to eq8_3_3_DRO {i in OSSERVAZIONI, j in FEATURES,
   t in LEAF_NODES, tt in A_L[t]}:
  z[i,t] == 1 ==> sum{k in 1..5}( sum{jj in FEATURES}
     D_x[j,jj,k]*pi_greco[i,jj,tt,k] ) >= -beta[i,j,tt];


subject to eq8_2 {i in OSSERVAZIONI, t in LEAF_NODES, a_r
   in A_R[t]}:
```

```
  if z[i,t] == 1 then sum {j in FEATURES}
     (a[j,a_r]*x[i,j]) >= b[a_r];

subject to eq8_4 {i in OSSERVAZIONI, t in LEAF_NODES, a_r
   in A_R[t]}:
  z[i,t] == 1 ==> -eta[i,a_r] -sum{jj in FEATURES}
     (beta[i,jj,a_r]*( y[i]*varrho_classe1[jj] +
     (1-y[i])*varrho_classe0[jj] ) ) >= b[a_r];

subject to eq8_4_1_DRO {i in OSSERVAZIONI, t in
   LEAF_NODES, tt in A_R[t]}:
  z[i,t] == 1 ==> sum{k in 1..5}( sum{jj in FEATURES}
     h_i[i,jj,k]*pi_greco[i,jj,tt,k] ) <= eta[i,tt];

subject to eq8_4_2_DRO {i in OSSERVAZIONI, j in FEATURES,
   t in LEAF_NODES, tt in A_R[t]}:
  z[i,t] == 1 ==> sum{k in 1..5}( sum{jj in FEATURES}
     (y[i]*C_x_classe1[j,jj,k] +
     (1-y[i])*C_x_classe0[j,jj,k] )*pi_greco[i,jj,tt,k] )
     >= -a[j,tt];

subject to eq8_4_3_DRO {i in OSSERVAZIONI, j in FEATURES,
   t in LEAF_NODES, tt in A_R[t]}:
  z[i,t] == 1 ==> sum{k in 1..5}( sum{jj in FEATURES}
     D_x[j,jj,k]*pi_greco[i,jj,tt,k] ) >= -beta[i,j,tt];
```

Come si può notare è stata introdotta l'istruzione condizionata *if* che prende come argomento la variabile *z[i,t]*, se questa è uguale a 1 allora il vincolo viene attivato, altrimenti viene semplicemente ignorato.

Per una visione totale del codice del modello distribuzionalmente robusto vedere la relativa sezione in appendice.

6.4.2 Risultati numerici

Al fine di testare la bontà del modello sono stati utilizzati gli indicatori (4.1). E' stato quindi eseguito il metodo della *K-fold cross validation* per diversi valori dei parametri ρ e K ottenendo i seguenti risultati:

Caso $\rho = 0.1$, $K = 1$:

1. *Mean out-sample accuracy*: **71.33%**.
2. *Richiamo*: **70.66%**.
3. *Precisione*: **81.76%**.

Caso $\rho = 0.01$, $K = 1$:

1. *Mean out-sample accuracy*: **66.00%**.
2. *Richiamo*: **64.00%**.
3. *Precisione*: **76.00%**.

Caso $\rho = 0.1$, $K = 2$:

1. *Mean out-sample accuracy*: **74.66%**.
2. *Richiamo*: **74.99%**.
3. *Precisione*: **76.66%**.

Caso $\rho = 0.01$, $K = 2$:

1. *Mean out-sample accuracy*: **78.00%**.
2. *Richiamo*: **76.32%**.
3. *Precisione*: **82.6%**.

Caso $\rho = 0.001$, $K = 2$:

1. *Mean out-sample accuracy*: **78.00%**.
2. *Richiamo*: **76.32%**.
3. *Precisione*: **82.60%**.

Caso $\rho = 0.0001$, $K = 2$:

1. *Mean out-sample accuracy*: **78.00%**.
2. *Richiamo*: **76.32%**.
3. *Precisione*: **82.60%**.

Caso $\rho = 0.001$, $K = 4$:

1. *Mean out-sample accuracy*: **78.00%**.

2. *Richiamo*: **76.32%**.

3. *Precisione*: **82.60%**.

In accordo con l'articolo "*Robust and Distributionally Robust Optimization Models for Linear Support Vector Machine*" [4] dei tre metodi robusti proposti il *distributionally robust* è quello che ha performato peggio in quanto la dimensione del *Dataset* è relativamente ridotta. Infatti come riportato nel l'articolo citato questo metodo performa meglio degli altri quando il numero delle osservazioni è almeno pari a 500, in quanto metodi meno conservativi permettono di comprendere meglio la struttura sottostante ai dati quando il *Dataset* ha un'elevata dimensionalità.

Conclusioni

	Deterministico	Robusto Iperettangolare	**Robusto Iperellissoidale**	Distribuzionalmente robusto
Accuratezza	83.5%	79.3%	**85.3%**	78.0%
Richiamo	86.0%	79.6%	**86.10%**	76.32%
Precisione	88.2%	83.5%	**92.33%**	82.6%

Tabella 6.1: Tabella riassuntiva risultati.

In questo lavoro di tesi è stato introdotto uno strumento matematico a supporto dei medici in grado di dare un'indicazione iniziale circa il decorso di un paziente affetto di COVID-19. I risultati migliori per ognuno dei metodi precedentemente esposti sono stati riassunti nella tabella 6.1 mentre il risultato migliore in assoluto è stato ottenuto con il modello a robustezza iperellissoidale con un'accuratezza dell' **85.33%** la quale può fornire un punto di partenza affidabile per definire il trattamento di un determinato paziente.

Facendo un confronto con quanto ottenuto in [10] dove è stato applicato il metodo di classificazione supervisionata *Support Vector Machine*, si può notare come si ottengano risultati migliori attraverso gli alberi decisionali. Questo non stupisce in quanto si può interpretare il *Support Vector Machine* come un caso particolare di *Decision Tree* dove come unico *Branch Node* si considera il nodo radice.

Un ulteriore confronto per valutare le prestazioni dell'algoritmo può essere fatto paragonandolo con gli algoritmi di *Machine Learning* inclusi nella *Classification Learner App* di Matlab tra i quali rientrano oltre agli alberi di decisioje: *Linear and Quadratic Discriminant Analysis*, *Logistic Regression*, *Naive Bayes*, *Support Vector Machine*, *k-Nearest Neighbors*, *Ensemble Classifier*.

L'algoritmo proposto da *Bertsimas e Dunn* [2] è stato qui migliorato per tener conto dell'incertezza dei dati con la tecnica a robustezza iperellissoidale ed è stato in grado di sovraperformare tutti gli algoritmi della *Classification Learner App* di Matlab ad eccezzione dell' *Optimizable Decision Tree* dove si è ottenuta un'accuratezza circa uguale (86.0%) [10].

E' importante sottolineare il fatto che i risultati ottenuti in questa tesi potrebbero essere ulteriormente migliorati in quanto per ognuno dei metodi proposti non si è mai giunti ad una soluzione ottima bensì l'algoritmo terminava sempre con una soluzione ammissibile in quanto si raggiungeva sempre il limite di tempo impostato. Questo inconveniente può essere facilmente superato sia aumentando il limite di tempo che usando una macchina con migliori capacità computazionali.

In conclusione nonostante gli ottimi risultati ottenuti bisogna comunque ricordare che questo strumento non sostituisce la valutazione del medico bensì è a supporto alla presa delle decisioni del medico e può essere utile utilizzarlo in situazioni di emergenza come quella dovuta al COVID-19 dove il Servizio Sanitario è stato soggetto a forti pressioni dovute al sovraffollamento degli ospedali.

Appendice

Optimal Classification Tree Multivariato

file *init.run:*

```
reset;
model optimalTree.mod;
data data0.dat;
option solver gurobi;
option gurobi_options "timelim=1200 IntegralityFocus=1";

solve;

param temp_val;
param t_val;
param nodo_foglia_calcolato {OSSERVAZIONI};
param classe_calcolata {OSSERVAZIONI};
param veri_positivi;
param veri_negativi;
param falsi_positivi;
param falsi_negativi;
param accuratezza;
param richiamo;
param precisione;

for{i in OSSERVAZIONI}{
  let nodo_foglia_calcolato[i] := 0;
  let classe_calcolata[i] := 0;
  }
```

```
#il modello considera uguali a zero tutte le variabili
   inferiori al parametro intFeastol(defaule 1e-05) di
   conseguenza se non sono esattamente uguali a zero nella
   fase del caclolo della classe potrebbe causare degli
   errori
for {p in FEATURES, t in BRANCH_NODES}{
     if d[t] = 0 then {
       let a[p,t] := 0;
       let b[t] := 0;
     }
   }

#ciclo che permette di calcolare il nodo foglia di
   destinazione di un'osservazione x(i)
for{i in OSSERVAZIONI}{
   let t_val := 1;
   repeat {
     let temp_val :=0.0;
     for{p in FEATURES}{
       let temp_val := temp_val + a[p, t_val] * x[i,p];
          }

     if temp_val < b[t_val] then{
       let nodo_foglia_calcolato[i] := t_val*2;
       let t_val := t_val*2;
     }else{
        let nodo_foglia_calcolato[i] := t_val*2 + 1;
        let t_val := t_val*2 + 1;
          }

     }while t_val <= floor(T/2) ;
```

```
   }

#ciclo che permette di calcolare la classe di
   un'osservazione x(i)
for{i in OSSERVAZIONI}{
   for{l_val in LEAF_NODES}{
      if c_kt[0, l_val] == 1 then{
         if nodo_foglia_calcolato[i] == l_val then
            let classe_calcolata[i] := 0;
      }
      if c_kt[1, l_val] == 1 then{
         if nodo_foglia_calcolato[i] == l_val then
            let classe_calcolata[i] := 1;
      }
   }
}

let veri_positivi := 0;
let veri_negativi := 0;
let falsi_positivi := 0;
let falsi_negativi := 0;
let accuratezza := 0;
let richiamo := 0;
let precisione := 0;

#calcola gli indicatori
for{i in OSSERVAZIONI}{

   if classe_calcolata[i] == 0 and y[i] == 0 then
     let veri_negativi := veri_negativi + 1;
```

```
    if classe_calcolata[i] == 1 and y[i] == 1 then
      let veri_positivi := veri_positivi + 1;

    if classe_calcolata[i] == 0 and y[i] == 1 then
      let falsi_negativi := falsi_negativi + 1;

    if classe_calcolata[i] == 1 and y[i] == 0 then
      let falsi_positivi := falsi_positivi + 1;

   }

if (veri_positivi + veri_negativi + falsi_positivi +
   falsi_negativi) != 0 then
   let accuratezza := (veri_positivi
      +veri_negativi)/(veri_positivi + veri_negativi +
      falsi_positivi + falsi_negativi);

if (veri_positivi + falsi_negativi) != 0 then
   let richiamo := veri_positivi / (veri_positivi +
      falsi_negativi);

if (veri_positivi + falsi_positivi) != 0 then
   let precisione := veri_positivi / (veri_positivi +
      falsi_positivi);


display accuratezza;
display a;
display a_capp;
display b;
display d;
```

```
display s;
display c_kt;
display L_t;
display z;
```

File *data.dat:*

```
let N := 150;
let P := 15;
let mu := 0.00005;
let alpha := 0.02;
let D := 3;
let N_min := 5;
let T := 2^(D+1) -1;

let OSSERVAZIONI := 1 .. N;
let FEATURES := 1 .. P;
let BRANCH_NODES := 1..floor(T/2);
let LEAF_NODES := floor(T/2) +1 .. T;
let TOTAL_NODES := BRANCH_NODES union LEAF_NODES;
let CLASSI := {0,1};

#inizilizzazione degli ancestor
for{t in TOTAL_NODES}{
let A_L[t] := {} ;
let A_R[t] := {} ;
}

#calcolo degli ancestor
for{t in TOTAL_NODES diff {1} }{
   let temp := t;
   repeat {
      if temp mod 2 == 0 then
       let A_L[t] := A_L[t] union {temp/2};
    else
       let A_R[t] := A_R[t] union {floor(temp/2)};
```

```
    let temp := floor(temp/2);
  } while temp/2 >=1 ;
}

#leggo i dati delle features e delle classi
read {i in OSSERVAZIONI, j in FEATURES} x[i,j] < x.txt;
read {i in OSSERVAZIONI} y[i] < y.txt;

param x_completo{OSSERVAZIONI, FEATURES};
read {i in OSSERVAZIONI, j in FEATURES} x_completo[i,j] <
   x2.txt;
param x_max {j in FEATURES} = max {m in OSSERVAZIONI}
   x_completo[m,j];
param x_min {j in FEATURES} = min {m in OSSERVAZIONI}
   x_completo[m,j];

#normalizzazione vettore osservazioni
for {i in OSSERVAZIONI, j in FEATURES}{
   let x[i,j] := (x[i,j] - x_min[j]) / (x_max[j] -
      x_min[j] );
}

#Inizializzazione L_tilda
if sum {i in OSSERVAZIONI : y[i] == 0} y[i] >= sum {i in
   OSSERVAZIONI : y[i] == 1} y[i] then
   let L_tilda := sum {i in OSSERVAZIONI : y[i] == 0} y[i];
else
   let L_tilda := sum {i in OSSERVAZIONI : y[i] == 1} y[i];

#calcolo a_l_max
for{t in LEAF_NODES diff {T}: t mod 2 != 0}{
```

```
  let a_l_max[t] := max{m in A_L[t]} m;
}
```

File *model.mod:*

```
#insiemi del modello
set OSSERVAZIONI;
set FEATURES;
set BRANCH_NODES;
set LEAF_NODES ;
set TOTAL_NODES;
set CLASSI;
set A_L {TOTAL_NODES};
set A_R {TOTAL_NODES};
set PERCORSO {LEAF_NODES};


#parametri del modello
param N;
param P;
param T;
param D;
param N_min;
param mu;
param alpha;
param L_tilda;
param temp;
param x {OSSERVAZIONI, FEATURES};
param y {OSSERVAZIONI} binary;
param Y {i in OSSERVAZIONI, k in CLASSI} = if y[i] == k
   then 1 else -1;
```

```
#variabili del modello
var z {OSSERVAZIONI, TOTAL_NODES} binary ;
var l {LEAF_NODES} binary;
var s {FEATURES, BRANCH_NODES} binary;
var d {BRANCH_NODES} binary;
var b {BRANCH_NODES};
var a {FEATURES, BRANCH_NODES} <= 1 , >= -1;
var a_capp {FEATURES, BRANCH_NODES} <= 1 , >= -1;
var bin {FEATURES, BRANCH_NODES} binary;
var N_kt {k in CLASSI,t in LEAF_NODES} = 1/2 * sum {i in
   OSSERVAZIONI}(1 + Y[i,k])*z[i,t];
var N_t {t in LEAF_NODES} = sum {i in OSSERVAZIONI} z[i,t];
var c_kt {k in CLASSI, t in LEAF_NODES} binary;
var L_t{t in LEAF_NODES} >=0;


#FUNZIONE OBIETTIVO
minimize obiettivo:
   (1/L_tilda) *sum {t in LEAF_NODES (L_t[t]) + alpha *
      sum{t in BRANCH_NODES, j in FEATURES} (s[j,t]);


#vincoli del modello
subject to eq1 {t in BRANCH_NODES diff {1}}:
  d[t] <= d[floor(t/2)];

subject to eq2_1 {t in BRANCH_NODES}:
  b[t] <= d[t];

subject to eq2_2 {t in BRANCH_NODES}:
  -d[t] <= b[t];
```

```
subject to eq3_1 {t in BRANCH_NODES}:
  sum{j in FEATURES} s[j,t] >= d[t];

subject to eq3_2 {j in FEATURES, t in BRANCH_NODES}:
  s[j,t] <= d[t];

subject to eq4_1 {j in FEATURES, t in BRANCH_NODES}:
  a[j,t] <= s[j,t];

subject to eq4_2 {j in FEATURES, t in BRANCH_NODES}:
  -s[j,t] <= a[j,t];

subject to eq5_1 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] >= a[j,t];

subject to eq5_2 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] >= -a[j,t];

subject to eq5_3 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] <= -a[j,t] + 2*(1-bin[j,t]);

subject to eq5_4 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] <= a[j,t] + 2*(bin[j,t]);

subject to eq5_5 {t in BRANCH_NODES}:
  sum {j in FEATURES} a_capp[j,t] <= d[t];

subject to eq6_1{t in LEAF_NODES}:
  sum {i in OSSERVAZIONI} z[i,t] >= N_min*l[t];
```

```
subject to eq6_2{i in OSSERVAZIONI, t in LEAF_NODES}:
  z[i,t] <= l[t];

subject to eq7 {i in OSSERVAZIONI}:
  sum {t in LEAF_NODES} z[i,t] = 1;

subject to eq8_1 {i in OSSERVAZIONI, t in LEAF_NODES, a_l
   in A_L[t]}:
  sum {j in FEATURES} (a[j,a_l]*x[i,j]) + mu <= b[a_l] +
     (2 + mu)*(1 - z[i,t]);

subject to eq8_2 {i in OSSERVAZIONI, t in LEAF_NODES, a_r
   in A_R[t]}:
  sum {j in FEATURES} (a[j,a_r]*x[i,j]) >= b[a_r] - 2*(1 -
     z[i,t]);

subject to eq9 {t in LEAF_NODES}:
  sum {k in CLASSI} c_kt[k,t] = l[t];

subject to eq10 {k in CLASSI, t in LEAF_NODES}:
  L_t[t] <= N_t[t] - N_kt[k,t] + N*c_kt[k,t];

subject to eq11 {k in CLASSI, t in LEAF_NODES}:
  L_t[t] >= N_t[t] - N_kt[k,t] - N*(1 - c_kt[k,t]);

subject to eq12_1 {t in LEAF_NODES: t mod 2 == 0}:
  l[t] = d[t/2];

subject to eq12_2 {t in LEAF_NODES diff {T} : t mod 2 !=
   0}:
  l[t] = d[a_l_max[t]];
```

Box Robust Optimal Classification Tree

File *init.run:*

```
#dati usati per l'allenamento (147 osservazioni)
param x {1..147, 1..15};
param y {1..147} binary;
param L_tilda;

#contiene le deviazioni standard
param std_x_classe1 {1..15};
param std_x_classe0 {1..15};

#contiene tutte le 150 osservazioni
param x_completo {1..150, 1..15};
#leggo i dati completi delle 150 osservazioni
read {i in 1..150, j in 1..15} x_completo[i,j] < x.txt;

#parametri usati per normalizzare le osservazioni
param x_max {j in 1..15} = max {m in 1..150}
   x_completo[m,j];
param x_min {j in 1..15} = min {m in 1..150}
   x_completo[m,j];

model optimalTree.mod;
data data.dat;
option solver GUROBI;
option gurobi_options "timelim=1000 integralityfocus=1
   intfeastol=1e-09 feastol=1e-09";
```

```
#insiemi e variabili per la fase di validazione (3
   osservazioni)
set OSSERVAZIONI_VAL := 1 .. 3;
set OSSERVAZIONI_COMPLETO := 1 .. 150;

param T_VAL := 2^(D+1) -1;
param temp_val;
param t_val;
param x_val {OSSERVAZIONI_VAL, FEATURES};
param y_val {OSSERVAZIONI_VAL} binary;
param nodo_foglia_calcolato {OSSERVAZIONI_VAL};
param classe_calcolata {OSSERVAZIONI_VAL};
param nodo_foglia_calcolato_in_sample {OSSERVAZIONI};
param classe_calcolata_in_sample {OSSERVAZIONI};
param veri_positivi;
param veri_negativi;
param falsi_positivi;
param falsi_negativi;
param accuratezza;
param accuratezza_media;
param accuratezza_media_in_sample;
param richiamo;
param precisione;
param sol {1..3, 0..49};
param solInSample {1..3,0..49};


for {k in 0..49}{

   print "inizio iterazione" & k;
```

```
#leggo i dati delle features, delle classi e della
   deviazione standard dell'allenamento

read {i in OSSERVAZIONI, j in FEATURES} x[i,j] <
   ('dati_allenamento\osservazioni\x' & k & '.txt');
read {i in OSSERVAZIONI} y[i] <
   ('dati_allenamento\classi\y' & k & '.txt');
read {j in FEATURES} std_x_classe1[j] <
   ('dati_allenamento\deviazione standard\std_classe1_'
   & k & '.txt');
read {j in FEATURES} std_x_classe0[j] <
   ('dati_allenamento\deviazione standard\std_classe0_'
   & k & '.txt');

#serve per normalizzare il vettore delle osservazioni
for {i in OSSERVAZIONI, j in FEATURES}{
   if (x_max[j] - x_min[j]) != 0 then
      let x[i,j] := (x[i,j] - x_min[j]) / (x_max[j] -
         x_min[j] );
}

#leggo i dati delle features e delle classi della
   validazione
read {i in OSSERVAZIONI_VAL, j in FEATURES} x_val[i,j]
   < ('dati_validazione\osservazioni\x' & k & '.txt');
read {i in OSSERVAZIONI_VAL} y_val[i] <
   ('dati_validazione\classi\y' & k & '.txt');

#normalizzo i dati della validazione
for {i in OSSERVAZIONI_VAL, j in FEATURES}{
   if (x_max[j] - x_min[j]) != 0 then
```

```
        let x_val[i,j] := (x_val[i,j] - x_min[j]) /
            (x_max[j] - x_min[j] );
}

 #calcolo L_tilda
 if sum {i in OSSERVAZIONI : y[i] == 0} y[i] >= sum {i
    in OSSERVAZIONI : y[i] == 1} y[i] then
    let L_tilda := sum {i in OSSERVAZIONI : y[i] == 0}
       y[i];
 else
    let L_tilda := sum {i in OSSERVAZIONI : y[i] == 1}
       y[i];

  solve;

 for {p in FEATURES, t in BRANCH_NODES}{
    if d[t] == 0 then {
       let a[p,t] := 0;
       let a_capp[p,t] := 0;
       let b[t] := 0;
    }
 }

 for{i in OSSERVAZIONI_VAL}{
    let nodo_foglia_calcolato[i] := 0;
    let classe_calcolata[i] := 0;
 }

 for{i in OSSERVAZIONI_VAL}{
    let t_val := 1;
    repeat {
```

```
            let temp_val := 0.0;
            for{p in FEATURES}{
              let temp_val := temp_val + a[p, t_val] *
                 x_val[i,p];
            }
            if temp_val < b[t_val] then{
              let nodo_foglia_calcolato[i] := t_val*2;
              let t_val := t_val*2;
            }else{
              let nodo_foglia_calcolato[i] := t_val*2 + 1;
              let t_val := t_val*2 + 1;
            }
         }while t_val <= floor(T/2) ;
      }

      for{i in OSSERVAZIONI_VAL}{}
         for{l_val in LEAF_NODES}{
            if c_kt[0, l_val] == 1 then{
               if nodo_foglia_calcolato[i] == l_val then
                  let classe_calcolata[i] := 0;
      }
            if c_kt[1, l_val] == 1 then{
               if nodo_foglia_calcolato[i] == l_val then
                  let classe_calcolata[i] := 1;
         }
      }
   }

      let veri_positivi := 0;
      let veri_negativi := 0;
      let falsi_positivi := 0;
```

```
let falsi_negativi := 0;
let accuratezza := 0;
let richiamo := 0;
let precisione := 0;

#calcolo degli indicatori
for{i in OSSERVAZIONI_VAL}{

if classe_calcolata[i] == 0 and y_val[i] == 0 then
   let veri_negativi := veri_negativi + 1;

if classe_calcolata[i] == 1 and y_val[i] == 1 then
   let veri_positivi := veri_positivi + 1;

if classe_calcolata[i] == 0 and y_val[i] == 1 then
   let falsi_negativi := falsi_negativi + 1;

if classe_calcolata[i] == 1 and y_val[i] == 0 then
   let falsi_positivi := falsi_positivi + 1;
}

if (veri_positivi + veri_negativi + falsi_positivi +
   falsi_negativi) != 0 then
   let accuratezza := (veri_positivi
      +veri_negativi)/(veri_positivi + veri_negativi +
      falsi_positivi + falsi_negativi);
if (veri_positivi + falsi_negativi) != 0 then
   let richiamo := veri_positivi / (veri_positivi +
      falsi_negativi);
if (veri_positivi + falsi_positivi) != 0 then
```

```
    let precisione := veri_positivi / (veri_positivi +
       falsi_positivi);

 let sol[1,k] := accuratezza;
 let sol[2,k] := richiamo;
 let sol[3,k] := precisione;

 print 'accuratezza out sample';
 display accuratezza;

 #fase in-sample
 for{i in OSSERVAZIONI}{
   let nodo_foglia_calcolato_in_sample[i] := 0;
   let classe_calcolata_in_sample[i] := 0;
}

 for{i in OSSERVAZIONI}{
 let t_val := 1;
 repeat {
        let temp_val := 0.0;
        for{p in FEATURES}{
          let temp_val := temp_val + a[p, t_val] *
             x[i,p];
        }
        if temp_val < b[t_val] then{
          let nodo_foglia_calcolato_in_sample[i] :=
             t_val*2;
          let t_val := t_val*2;
        }else{
          let nodo_foglia_calcolato_in_sample[i] :=
             t_val*2 + 1;
```

```
            let t_val := t_val*2 + 1;
        }
    }while t_val <= floor(T/2) ;
}

for{i in OSSERVAZIONI}{
    for{l_val in LEAF_NODES}{
        if c_kt[0, l_val] == 1 then{
            if nodo_foglia_calcolato_in_sample[i] == l_val
                then
                let classe_calcolata_in_sample[i] := 0;
        }

        if c_kt[1, l_val] == 1 then{
            if nodo_foglia_calcolato_in_sample[i] == l_val
                then
                let classe_calcolata_in_sample[i] := 1;
        }
    }
}

let veri_positivi := 0;
let veri_negativi := 0;
let falsi_positivi := 0;
let falsi_negativi := 0;
let accuratezza := 0;
let richiamo := 0;
let precisione := 0;

#calcolo degli indicatori
for{i in OSSERVAZIONI}{
```

```
    if classe_calcolata_in_sample[i] == 0 and y[i] == 0
       then
        let veri_negativi := veri_negativi + 1;

    if classe_calcolata_in_sample[i] == 1 and y[i] == 1
       then
        let veri_positivi := veri_positivi + 1;

    if classe_calcolata_in_sample[i] == 0 and y[i] == 1
       then
        let falsi_negativi := falsi_negativi + 1;

    if classe_calcolata_in_sample[i] == 1 and y[i] == 0
       then
        let falsi_positivi := falsi_positivi + 1;

 }

 if (veri_positivi + veri_negativi + falsi_positivi +
    falsi_negativi) != 0 then
   let accuratezza := (veri_positivi
      +veri_negativi)/(veri_positivi + veri_negativi +
      falsi_positivi + falsi_negativi);
if (veri_positivi + falsi_negativi) != 0 then
   let richiamo := veri_positivi / (veri_positivi +
      falsi_negativi);
if (veri_positivi + falsi_positivi) != 0 then
   let precisione := veri_positivi / (veri_positivi +
      falsi_positivi);
```

```
let solInSample[1,k] := accuratezza;
let solInSample[2,k] := richiamo;
let solInSample[3,k] := precisione;

print 'accuratezza_in sample';
display accuratezza;
display a;
display a_capp;
display b;
display d;
display s;
display c_kt;
display L_t;
display N_t;
display N_kt;
display nodo_foglia_calcolato;
display classe_calcolata;

print "fine iterazione" & k;

reset data z;
reset data l;
reset data s;
reset data d;
reset data b;
reset data a;
reset data a_capp;
reset data N_kt;
reset data N_t;
reset data c_kt;
reset data L_t;
```

```
  reset data x;
  reset data std_x_classe1;
  reset data std_x_classe0;
  reset data y;
  reset data x_val;
  reset data y_val;
}

let accuratezza_media := 0;
for{j in 0..49}{
  let accuratezza_media := accuratezza_media + sol[1,j];
}
let accuratezza_media := accuratezza_media / 50;

let accuratezza_media_in_sample := 0;
for{j in 0..49}{
  let accuratezza_media_in_sample :=
     accuratezza_media_in_sample + solInSample[1,j];
}
let accuratezza_media_in_sample :=
   accuratezza_media_in_sample / 50;
```

File *data.dat*

```
let N := 147;
let P := 15;
let mu := 0.0000005;
let alpha := 0.02;
let D := 2;
let N_min := 5;
let T := 2^(D+1) -1;
let ro := 0.00001;

let OSSERVAZIONI := 1 .. N;
let FEATURES := 1 .. P;
let BRANCH_NODES := 1..floor(T/2);
let LEAF_NODES := floor(T/2) +1 .. T;
let TOTAL_NODES := BRANCH_NODES union LEAF_NODES;
let CLASSI := {0,1};

#inizilizzazione degli ancestor
for{t in TOTAL_NODES}{
let A_L[t] := {} ;
let A_R[t] := {} ;
}

#calcolo degli ancestor
for{t in TOTAL_NODES diff {1} }{
let temp := t;
repeat {
   if temp mod 2 == 0 then
      let A_L[t] := A_L[t] union {temp/2};
      else
```

```
      let A_R[t] := A_R[t] union {floor(temp/2)};
   let temp := floor(temp/2);
   } while temp/2 >=1 ;
}

#calcolo a_l_max
for{t in LEAF_NODES diff {T}: t mod 2 != 0}{
  let a_l_max[t] := max{m in A_L[t]} m;
}
```

File *model.mod:*

```
#insiemi del modello
set OSSERVAZIONI;
set FEATURES;
set BRANCH_NODES;
set LEAF_NODES ;
set TOTAL_NODES;
set CLASSI;
set A_L {TOTAL_NODES};
set A_R {TOTAL_NODES};

#parametri del modello
param ro;
param N;
param P;
param T;
param D;
param N_min;
param mu;
param alpha;
param temp;
param Y {i in OSSERVAZIONI, k in CLASSI} = if y[i] == k
   then 1 else -1;

#variabili del modello
var z {OSSERVAZIONI, TOTAL_NODES} binary ;
var l {LEAF_NODES} binary;
var s {FEATURES, BRANCH_NODES} binary;
var d {BRANCH_NODES} binary;
var b {BRANCH_NODES} <= 1 , >= -1;
```

```
var a {FEATURES, BRANCH_NODES} <= 1 , >= -1;
var a_capp {FEATURES, BRANCH_NODES} <= 1 , >= -1;
var bin {FEATURES, BRANCH_NODES} binary;
var N_kt {k in CLASSI,t in LEAF_NODES} = 1/2 * sum {i in
   OSSERVAZIONI}(1 + Y[i,k])*z[i,t];
var N_t {t in LEAF_NODES} = sum {i in OSSERVAZIONI} z[i,t];
var c_kt {k in CLASSI, t in LEAF_NODES} binary;
var L_t{t in LEAF_NODES} >=0;


#FUNZIONE OBIETTIVO
minimize obiettivo:
   (1/L_tilda) *sum {t in LEAF_NODES} (L_t[t])
     + alpha * sum{t in BRANCH_NODES, j in FEATURES}
        (s[j,t]);


#vincoli del modello
subject to eq1 {t in BRANCH_NODES diff {1}}:
  d[t] <= d[floor(t/2)];

subject to eq2_1 {t in BRANCH_NODES}:
  b[t] <= d[t];

subject to eq2_2 {t in BRANCH_NODES}:
  -d[t] <= b[t];

subject to eq3_1 {t in BRANCH_NODES}:
  sum{j in FEATURES} s[j,t] >= d[t];

subject to eq3_2 {j in FEATURES, t in BRANCH_NODES}:
```

```
  s[j,t] <= d[t];

subject to eq4_1 {j in FEATURES, t in BRANCH_NODES}:
  a[j,t] <= s[j,t];

subject to eq4_2 {j in FEATURES, t in BRANCH_NODES}:
  -s[j,t] <= a[j,t];

subject to eq5_1 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] >= a[j,t];

subject to eq5_2 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] >= -a[j,t];

subject to eq5_3 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] <= -a[j,t] + 2*(1-bin[j,t]);

subject to eq5_4 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] <= a[j,t] + 2*(bin[j,t]);

subject to eq5_5 {t in BRANCH_NODES}:
  sum {j in FEATURES} a_capp[j,t] <= d[t];

subject to eq6_1{t in LEAF_NODES}:
  sum {i in OSSERVAZIONI} z[i,t] >= N_min*l[t];

subject to eq6_2{i in OSSERVAZIONI, t in LEAF_NODES}:
  z[i,t] <= l[t];

subject to eq7 {i in OSSERVAZIONI}:
  sum {t in LEAF_NODES} z[i,t] = 1;
```

```
subject to eq8_1 {i in OSSERVAZIONI, t in LEAF_NODES, a_l
   in A_L[t]}:
  sum {j in FEATURES} (a[j,a_l]*x[i,j]) + ro*(sum {j in
     FEATURES} a_capp[j,a_l]*( std_x_classe1[j]*(y[i]) +
     std_x_classe0[j]*(1 - y[i]) )) + mu <= b[a_l] + (3 +
     mu)*(1 - z[i,t]);

subject to eq8_2 {i in OSSERVAZIONI, t in LEAF_NODES, a_r
   in A_R[t]}:
  sum {j in FEATURES} (a[j,a_r]*x[i,j]) - ro*(sum {j in
     FEATURES} a_capp[j,a_r]*( std_x_classe1[j]*(y[i]) +
     std_x_classe0[j]*(1 - y[i]) )) >= b[a_r] - 3*(1 -
     z[i,t]);

subject to eq9 {t in LEAF_NODES}:
  sum {k in CLASSI} c_kt[k,t] = l[t];

subject to eq10 {k in CLASSI, t in LEAF_NODES}:
  L_t[t] <= N_t[t] - N_kt[k,t] + N*c_kt[k,t];

subject to eq11 {k in CLASSI, t in LEAF_NODES}:
  L_t[t] >= N_t[t] - N_kt[k,t] - N*(1 - c_kt[k,t]);

subject to eq12_1 {t in LEAF_NODES: t mod 2 == 0}:
  l[t] = d[t/2];

subject to eq12_2 {t in LEAF_NODES diff {T} : t mod 2 !=
   0}:
  l[t] = d[a_l_max[t]];
```

Ellipsoidal Robust Optimal Classification Tree

File *init.run:*

```
#dati usati per l'allenamento (147 osservazioni)
param x {1..147, 1..15};
param y {1..147} binary;

#serve per normalizzare L_t
param L_tilda;

#deviazioni standard
param std_x_classe1 {1..15};
param std_x_classe0 {1..15};

#matrice covarianza
param cov_matrix_classe1 {1..15, 1..15};
param cov_matrix_classe0 {1..15, 1..15};

#contiene tutte le 150 osservazioni
param x_completo {1..150, 1..15};
#leggo i dati delle 150 osservazioni
read {i in 1..150, j in 1..15} x_completo[i,j] < x.txt;

#parametri usati per normalizzare i dati
param x_max {j in 1..15} = max {m in 1..150}
   x_completo[m,j];
param x_min {j in 1..15} = min {m in 1..150}
   x_completo[m,j];
```

```
model optimalTree.mod;
data data.dat;
option solver gurobi;
option gurobi_options "timelim=1000 integralityfocus=1
   intfeastol=1e-07 feastol=1e-07";

#insiemi e variabili per la fase di validazione
set OSSERVAZIONI_VAL := 1 .. 3;
set OSSERVAZIONI_COMPLETO := 1 .. 150;

param T_VAL := 2^(D+1) -1;
param temp_val;
param t_val;
param x_val {OSSERVAZIONI_VAL, FEATURES};
param y_val {OSSERVAZIONI_VAL} binary;
param nodo_foglia_calcolato {OSSERVAZIONI_VAL};
param classe_calcolata {OSSERVAZIONI_VAL};
param nodo_foglia_calcolato_in_sample {OSSERVAZIONI};
param classe_calcolata_in_sample {OSSERVAZIONI};
param veri_positivi;
param veri_negativi;
param falsi_positivi;
param falsi_negativi;
param accuratezza;
param accuratezza_media;
param accuratezza_media_in_sample;
param richiamo;
param precisione;
param sol {1..3, 0..49};
param solInSample {1..3,0..49};
```

```
for {k in 0..49}{

   print "inizio iterazione" & k;

   #leggo i dati delle features, delle classi e delle
      deviazioni standard dell'allenamento
   read {i in OSSERVAZIONI, j in FEATURES} x[i,j] <
      ('dati_allenamento\osservazioni\x' & k & '.txt');
   read {i in OSSERVAZIONI} y[i] <
      ('dati_allenamento\classi\y' & k & '.txt');
   read {j in FEATURES} std_x_classe1[j] <
      ('dati_allenamento\deviazione standard\std_classe1_'
      & k & '.txt');
   read {j in FEATURES} std_x_classe0[j] <
      ('dati_allenamento\deviazione standard\std_classe0_'
      & k & '.txt');

   #normalizzazione dati
   for {i in OSSERVAZIONI, j in FEATURES}{
    if (x_max[j] - x_min[j]) != 0 then
     let x[i,j] := (x[i,j] - x_min[j]) / (x_max[j] -
        x_min[j] );
  }

   #calcolo matrice COVARIANZA
   for{j in FEATURES, jj in FEATURES}{
      if(j == jj) then
         let cov_matrix_classe1[j,jj] :=
            std_x_classe1[j]*std_x_classe1[j];
      else
```

```
        let cov_matrix_classe1[j,jj] := 0;
}

for{j in FEATURES, jj in FEATURES}{
   if(j == jj) then
        let cov_matrix_classe0[j,jj] :=
           std_x_classe0[j]*std_x_classe0[j];
   else
        let cov_matrix_classe0[j,jj] := 0;
}


#leggo i dati delle features e delle classi della
   validazione
read {i in OSSERVAZIONI_VAL, j in FEATURES} x_val[i,j]
   < ('dati_validazione\osservazioni\x' & k & '.txt');
read {i in OSSERVAZIONI_VAL} y_val[i] <
   ('dati_validazione\classi\y' & k & '.txt');

#normalizzo i dati della validazione
for {i in OSSERVAZIONI_VAL, j in FEATURES}{
if (x_max[j] - x_min[j]) != 0 then
  let x_val[i,j] := (x_val[i,j] - x_min[j]) / (x_max[j]
     - x_min[j] );
}

#calcolo L_tilda
if sum {i in OSSERVAZIONI : y[i] == 0} y[i] >= sum {i
   in OSSERVAZIONI : y[i] == 1} y[i] then
   let L_tilda := sum {i in OSSERVAZIONI : y[i] == 0}
      y[i];
```

```
else
   let L_tilda := sum {i in OSSERVAZIONI : y[i] == 1}
      y[i];

  solve;

for {p in FEATURES, t in BRANCH_NODES}{
   if d[t] == 0 then {
      let a[p,t] := 0;
      let a_capp[p,t] := 0;
      let b[t] := 0;
   }
   if abs(a[p,t]) < 1e-07 then {
      let a[p,t] := 0;
      let a_capp[p,t] := 0;
   }
   if abs(b[t]) < 1e-07 then {
      let b[t] := 0;
      }
}

for{i in OSSERVAZIONI_VAL}{
   let nodo_foglia_calcolato[i] := 0;
   let classe_calcolata[i] := 0;
}

for{i in OSSERVAZIONI_VAL}{
   let t_val := 1;
   repeat {
       let temp_val := 0.0;
       for{p in FEATURES}{
```

```
            let temp_val := temp_val + a[p, t_val] *
                x_val[i,p];
        }

        if temp_val < b[t_val] then{
            let nodo_foglia_calcolato[i] := t_val*2;
            let t_val := t_val*2;
        }else{
            let nodo_foglia_calcolato[i] := t_val*2 + 1;
            let t_val := t_val*2 + 1;
        }
      }while t_val <= floor(T/2) ;
}



for{i in OSSERVAZIONI_VAL}{
   for{l_val in LEAF_NODES}{
      if c_kt[0, l_val] == 1 then{
         if nodo_foglia_calcolato[i] == l_val then
            let classe_calcolata[i] := 0;
      }
      if c_kt[1, l_val] == 1 then{
         if nodo_foglia_calcolato[i] == l_val then
            let classe_calcolata[i] := 1;
      }
   }
}

let veri_positivi := 0;
let veri_negativi := 0;
```

```
let falsi_positivi := 0;
let falsi_negativi := 0;
let accuratezza := 0;
let richiamo := 0;
let precisione := 0;


#calcolo degli indicatori
for{i in OSSERVAZIONI_VAL}{

   if classe_calcolata[i] == 0 and y_val[i] == 0 then
      let veri_negativi := veri_negativi + 1;

   if classe_calcolata[i] == 1 and y_val[i] == 1 then
      let veri_positivi := veri_positivi + 1;

   if classe_calcolata[i] == 0 and y_val[i] == 1 then
      let falsi_negativi := falsi_negativi + 1;

   if classe_calcolata[i] == 1 and y_val[i] == 0 then
      let falsi_positivi := falsi_positivi + 1;

}

if (veri_positivi + veri_negativi + falsi_positivi +
   falsi_negativi) != 0 then
   let accuratezza := (veri_positivi
      +veri_negativi)/(veri_positivi + veri_negativi +
      falsi_positivi + falsi_negativi);

if (veri_positivi + falsi_negativi) != 0 then
```

```
   let richiamo := veri_positivi / (veri_positivi +
      falsi_negativi);

if (veri_positivi + falsi_positivi) != 0 then
   let precisione := veri_positivi / (veri_positivi +
      falsi_positivi);

let sol[1,k] := accuratezza;
let sol[2,k] := richiamo;
let sol[3,k] := precisione;

print 'accuratezza out sample';
display accuratezza;

#fase in-sample
for{i in OSSERVAZIONI}{
   let nodo_foglia_calcolato_in_sample[i] := 0;
   let classe_calcolata_in_sample[i] := 0;
}

for{i in OSSERVAZIONI}{

   let t_val := 1;
   repeat {
          let temp_val := 0.0;
          for{p in FEATURES}{
            let temp_val := temp_val + a[p, t_val] *
               x[i,p];
          }
          if temp_val < b[t_val] then{
```

```
            let nodo_foglia_calcolato_in_sample[i] :=
               t_val*2;
            let t_val := t_val*2;
          }else{
            let nodo_foglia_calcolato_in_sample[i] :=
               t_val*2 + 1;
            let t_val := t_val*2 + 1;
          }
      }while t_val <= floor(T/2) ;
}

for{i in OSSERVAZIONI}{
   for{l_val in LEAF_NODES}{
      if c_kt[0, l_val] == 1 then{
         if nodo_foglia_calcolato_in_sample[i] == l_val
            then
            let classe_calcolata_in_sample[i] := 0;
      }
      if c_kt[1, l_val] == 1 then{
         if nodo_foglia_calcolato_in_sample[i] == l_val
            then
            let classe_calcolata_in_sample[i] := 1;
      }
   }
}

let veri_positivi := 0;
let veri_negativi := 0;
let falsi_positivi := 0;
let falsi_negativi := 0;
let accuratezza := 0;
```

```
let richiamo := 0;
let precisione := 0;

#calcolo degli indicatori
for{i in OSSERVAZIONI}{

   if classe_calcolata_in_sample[i] == 0 and y[i] == 0
      then
      let veri_negativi := veri_negativi + 1;

   if classe_calcolata_in_sample[i] == 1 and y[i] == 1
      then
      let veri_positivi := veri_positivi + 1;

   if classe_calcolata_in_sample[i] == 0 and y[i] == 1
      then
      let falsi_negativi := falsi_negativi + 1;

   if classe_calcolata_in_sample[i] == 1 and y[i] == 0
      then
      let falsi_positivi := falsi_positivi + 1;

}

if (veri_positivi + veri_negativi + falsi_positivi +
   falsi_negativi) != 0 then
   let accuratezza := (veri_positivi
      +veri_negativi)/(veri_positivi + veri_negativi +
      falsi_positivi + falsi_negativi);

if (veri_positivi + falsi_negativi) != 0 then
```

```
   let richiamo := veri_positivi / (veri_positivi +
      falsi_negativi);

if (veri_positivi + falsi_positivi) != 0 then
   let precisione := veri_positivi / (veri_positivi +
      falsi_positivi);

let solInSample[1,k] := accuratezza;
let solInSample[2,k] := richiamo;
let solInSample[3,k] := precisione;

print 'accuratezza_in sample';
display accuratezza;
display a;
display b;
display d;
display c_kt;
display L_t;
display nodo_foglia_calcolato;
display classe_calcolata;

print "fine iterazione" & k;

reset data z;
reset data l;
reset data d;
reset data b;
reset data a;
reset data N_kt;
reset data N_t;
reset data c_kt;
```

```
    reset data L_t;
    reset data x;
    reset data y;
    reset data x_val;
    reset data y_val;
    reset data std_x_classe1;
    reset data std_x_classe0;
}

let accuratezza_media := 0;
for{j in 0..49}{
  let accuratezza_media := accuratezza_media + sol[1,j];
}
let accuratezza_media := accuratezza_media / 50;

let accuratezza_media_in_sample := 0;
for{j in 0..49}{
  let accuratezza_media_in_sample :=
     accuratezza_media_in_sample + solInSample[1,j];
}
let accuratezza_media_in_sample :=
   accuratezza_media_in_sample / 50;
```

File *data.dat:*

```
let N := 147;
let P := 15;
let mu := 0.000005;
let alpha := 0.02;
let D := 2;
let N_min := 5;
let T := 2^(D+1) -1;
let ro := 0.00002;

let OSSERVAZIONI := 1 .. N;
let FEATURES := 1 .. P;
let BRANCH_NODES := 1..floor(T/2);
let LEAF_NODES := floor(T/2) +1 .. T;
let TOTAL_NODES := BRANCH_NODES union LEAF_NODES;
let CLASSI := {0,1};

#inizilizzazione degli ancestor
for{t in TOTAL_NODES}{
let A_L[t] := {} ;
let A_R[t] := {} ;
}

#calcolo degli ancestor
for{t in TOTAL_NODES diff {1} }{
let temp := t;
  repeat {
    if temp mod 2 == 0 then
      let A_L[t] := A_L[t] union {temp/2};
      else
```

```
      let A_R[t] := A_R[t] union {floor(temp/2)};
    let temp := floor(temp/2);
    } while temp/2 >=1 ;
}
}

#calcolo a_l_max
for{t in LEAF_NODES diff {T}: t mod 2 != 0}{
  let a_l_max[t] := max{m in A_L[t]} m;
}
```

File *model.mod:*

```
#insiemi del modello
set OSSERVAZIONI;
set FEATURES;
set BRANCH_NODES;
set LEAF_NODES ;
set TOTAL_NODES;
set CLASSI;
set A_L {TOTAL_NODES};
set A_R {TOTAL_NODES};

#parametri del modello
param N;
param P;
param T;
param D;
param N_min;
param mu;
param alpha;
param temp;
param Y {i in OSSERVAZIONI, k in CLASSI} = if y[i] == k
   then 1 else -1;
param ro;

#variabili del modello
var z {OSSERVAZIONI, TOTAL_NODES} binary ;
var l {LEAF_NODES} binary;
var s {FEATURES, BRANCH_NODES} binary;
var d {BRANCH_NODES} binary;
var b {BRANCH_NODES} <= 1 , >= -1;
```

```
var a {FEATURES, BRANCH_NODES} <= 1 , >= -1;
var a_capp {FEATURES, BRANCH_NODES} <= 1 , >= -1;
var bin {FEATURES, BRANCH_NODES} binary;
var N_kt {k in CLASSI,t in LEAF_NODES} = 1/2 * sum {i in
   OSSERVAZIONI}(1 + Y[i,k])*z[i,t];
var N_t {t in LEAF_NODES} = sum {i in OSSERVAZIONI} z[i,t];
var c_kt {k in CLASSI, t in LEAF_NODES} binary;
var L_t{t in LEAF_NODES} >=0;


#FUNZIONE OBIETTIVO
minimize obiettivo:
   (1/L_tilda) * (sum {t in LEAF_NODES} L_t[t])
     + alpha * (sum{t in BRANCH_NODES, j in FEATURES}
        s[j,t]);

#vincoli del modello
subject to eq1 {t in BRANCH_NODES diff {1}}:
  d[t] <= d[floor(t/2)];

subject to eq2_1 {t in BRANCH_NODES}:
  b[t] <= d[t];

subject to eq2_2 {t in BRANCH_NODES}:
  -d[t] <= b[t];

subject to eq3_1 {t in BRANCH_NODES}:
  sum{j in FEATURES} s[j,t] >= d[t];

subject to eq3_2 {j in FEATURES, t in BRANCH_NODES}:
  s[j,t] <= d[t];
```

```
subject to eq4_1 {j in FEATURES, t in BRANCH_NODES}:
  a[j,t] <= s[j,t];

subject to eq4_2 {j in FEATURES, t in BRANCH_NODES}:
  -s[j,t] <= a[j,t];

subject to eq5_1 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] >= a[j,t];

subject to eq5_2 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] >= -a[j,t];

subject to eq5_3 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] <= -a[j,t] + 2*(1-bin[j,t]);

subject to eq5_4 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] <= a[j,t] + 2*(bin[j,t]);

subject to eq5_5 {t in BRANCH_NODES}:
  sum {j in FEATURES} a_capp[j,t] <= d[t];

subject to eq6_1{t in LEAF_NODES}:
  sum {i in OSSERVAZIONI} z[i,t] >= N_min*l[t];

subject to eq6_2{i in OSSERVAZIONI, t in LEAF_NODES}:
  z[i,t] <= l[t];

subject to eq7 {i in OSSERVAZIONI}:
  sum {t in LEAF_NODES} z[i,t] = 1;
```

```
subject to eq8_1 {i in OSSERVAZIONI, t in LEAF_NODES, a_l
   in A_L[t]}:
  sum {j in FEATURES} ( a[j,a_l]*x[i,j] ) + ro*sqrt( sum
     {j in FEATURES} a[j,a_l]*(
     cov_matrix_classe1[j,j]*y[i] +
     cov_matrix_classe0[j,j]*(1-y[i]) )*a[j,a_l] ) + mu <=
     b[a_l] + (3 + mu)*(1 - z[i,t]);

subject to eq8_2 {i in OSSERVAZIONI, t in LEAF_NODES, a_r
   in A_R[t]}:
  sum {j in FEATURES} (a[j,a_r]*x[i,j]) - ro*sqrt( sum {j
     in FEATURES} a[j,a_r]*( cov_matrix_classe1[j,j]*y[i]
     + cov_matrix_classe0[j,j]*(1-y[i]) )*a[j,a_r] ) >=
     b[a_r] - 3*(1 - z[i,t]);

subject to eq9 {t in LEAF_NODES}:
  sum {k in CLASSI} c_kt[k,t] = l[t];

subject to eq10 {k in CLASSI, t in LEAF_NODES}:
  L_t[t] <= N_t[t] - N_kt[k,t] + N*c_kt[k,t];

subject to eq11 {k in CLASSI, t in LEAF_NODES}:
  L_t[t] >= N_t[t] - N_kt[k,t] - N*(1 - c_kt[k,t]);

subject to eq12_1 {t in LEAF_NODES: t mod 2 == 0}:
  l[t] = d[t/2];

subject to eq12_2 {t in LEAF_NODES diff {T} : t mod 2 !=
   0}:
  l[t] = d[a_l_max[t]];
```

Distributionally Robust Optimal Classification Tree

File *init.run:*

```
#dati usati per l'allenamento (147 osservazioni)
param x {1..147, 1..15};
param y {1..147} binary;

#serve per normalizzare L_t
param L_tilda;

#parametro deviazioni standard
param std_x_classe1 {1..15};
param std_x_classe0 {1..15};

#parametro controllo perturbazione
param rho = 0.3;

#parametro soglia E_p[vettore rho]<= varrho
param varrho_classe1 {1..15};
param varrho_classe0 {1..15};

#parametro che serve a controllare il grado di
   conservatorismo
param K = 2;

#parametro h_i
param h_i {1..147, 1..15, 1..5};

#parametro C_x
param C_x_classe1 {1..15, 1..15, 1..5};
```

```
param C_x_classe0 {1..15, 1..15, 1..5};

#parametro D_x
param D_x {1..15, 1..15, 1..5};

#matrice contenente le componenti principali
param F_x_classe1 {1..15, 1..15};
param F_x_classe0 {1..15, 1..15};

#parametro contenente gli autovalori delle componenti
   principali ovvero la loro varianza
param lambda_classe1 {1..15};
param lambda_classe0 {1..15};

model optimalTree.mod;
data data.dat;
option solver gurobi;
option gurobi_options "timelim=1000 integralityfocus=1 ";


#insiemi e variabili per la fase di validazione (3
   osservazioni)
set OSSERVAZIONI_VAL := 1 .. 3;
set OSSERVAZIONI_COMPLETO := 1 .. 150;

param T_VAL := 2^(D+1) -1;
param temp_val;
param t_val;
param x_val {OSSERVAZIONI_VAL, FEATURES};
param y_val {OSSERVAZIONI_VAL} binary;
param nodo_foglia_calcolato {OSSERVAZIONI_VAL};
```

```
param classe_calcolata {OSSERVAZIONI_VAL};
param nodo_foglia_calcolato_in_sample {OSSERVAZIONI};
param classe_calcolata_in_sample {OSSERVAZIONI};
param veri_positivi;
param veri_negativi;
param falsi_positivi;
param falsi_negativi;
param accuratezza;
param accuratezza_media;
param accuratezza_media_in_sample;
param richiamo;
param precisione;
param sol {1..3, 0..49};
param solInSample {1..3,0..49};
param temp2;


for {k in 0..49}{

   print "inizio iterazione" & k;

   #leggo i dati delle features, delle classi e delle
      deviazioni standard dell'allenamento
  read {i in OSSERVAZIONI, j in FEATURES} x[i,j] <
     ('dati_allenamento\osservazioni\x' & k & '.txt');
  read {i in OSSERVAZIONI} y[i] <
     ('dati_allenamento\classi\y' & k & '.txt');
  read {j in FEATURES} std_x_classe1 [j] <
     ('dati_allenamento\deviazione
     standard\std_x_classe1_' & k & '.txt');
```

```
read {j in FEATURES} std_x_classe0 [j] <
   ('dati_allenamento\deviazione
   standard\std_x_classe0_' & k & '.txt');
read {j in FEATURES} lambda_classe1 [j] <
   ('dati_allenamento\varianza componenti
   principali\lambda_classe1_' & k & '.txt');
read {j in FEATURES} lambda_classe0 [j] <
   ('dati_allenamento\varianza componenti
   principali\lambda_classe0_' & k & '.txt');
read {i in FEATURES, j in FEATURES} F_x_classe1 [i,j] <
   ('dati_allenamento\componenti principali\pc_classe1_'
   & k & '.txt');
read {i in FEATURES, j in FEATURES} F_x_classe0 [i,j] <
   ('dati_allenamento\componenti principali\pc_classe0_'
   & k & '.txt');

 #costruzione parametro varrho
 for {j in FEATURES}{
    let varrho_classe1[j] := (
       rho*sqrt(lambda_classe1[j]) )/K;
}

 #costruzione parametro sigma
 for {j in FEATURES}{
    let varrho_classe0[j] := (
       rho*sqrt(lambda_classe0[j]) )/K;
}

 #costruzione parametro h_i
 for {i in OSSERVAZIONI, j in FEATURES}{
```

```
    let h_i[i,j,1] := x[i,j] + rho*(
       y[i]*std_x_classe1[j] + (1-y[i])*std_x_classe0[j]
       );
    let h_i[i,j,2] := -x[i,j] + rho*(
       y[i]*std_x_classe1[j] + (1-y[i])*std_x_classe0[j]
       );
    let h_i[i,j,3] := 0;

    let temp2 := 0;

    #questo ciclo for rappresenta il prodotto scalare
       del vettore x con una componente
    for {jj in FEATURES}{
       let temp2 := temp2 + x[i,jj]*(
          y[i]*F_x_classe1[jj,j] +
          (1-y[i])*F_x_classe0[jj,j] );
  }

       let h_i[i,j,4] := temp2;
       let h_i[i,j,5] := -temp2;
}

 #costruzione parametro C_x
 for {i in FEATURES, j in FEATURES}{

    if i == j then
       let C_x_classe1[i,j,1] := 1;
    else
       let C_x_classe1[i,j,1] := 0;

    if i == j then
```

```
        let C_x_classe1[i,j,2] := -1;
    else
        let C_x_classe1[i,j,2] := 0;

    let C_x_classe1[i,j,3] := 0;

    let C_x_classe1[i,j,4] := F_x_classe1[j,i];

    let C_x_classe1[i,j,5] := -F_x_classe1[j,i];
}

 for {i in FEATURES, j in FEATURES}{

  if i == j then
    let C_x_classe0[i,j,1] := 1;
  else
    let C_x_classe0[i,j,1] := 0;

  if i == j then
    let C_x_classe0[i,j,2] := -1;
  else
    let C_x_classe0[i,j,2] := 0;

  let C_x_classe0[i,j,3] := 0;

  let C_x_classe0[i,j,4] := F_x_classe0[j,i];

  let C_x_classe0[i,j,5] := -F_x_classe0[j,i];
}

 #costruzione parametro D
```

```
for {i in FEATURES, j in FEATURES}{

   let D_x[i,j,1] := 0;

   let D_x[i,j,2] := 0;

   if i == j then
      let D_x[i,j,3] := -1;
   else
      let D_x[i,j,3] := 0;

   if i == j then
      let D_x[i,j,4] := -1;
   else
      let D_x[i,j,4] := 0;

   if i == j then
      let D_x[i,j,5] := -1;
   else
      let D_x[i,j,5] := 0;
}

#leggo i dati delle features e delle classi della
   validazione
read {i in OSSERVAZIONI_VAL, j in FEATURES} x_val[i,j]
   < ('dati_validazione\osservazioni\x' & k & '.txt');
read {i in OSSERVAZIONI_VAL} y_val[i] <
   ('dati_validazione\classi\y' & k & '.txt');

#calcolo L_tilda
```

```
if sum {i in OSSERVAZIONI : y[i] == 0} y[i] >= sum {i
   in OSSERVAZIONI : y[i] == 1} y[i] then
   let L_tilda := sum {i in OSSERVAZIONI : y[i] == 0}
      y[i];
else
   let L_tilda := sum {i in OSSERVAZIONI : y[i] == 1}
      y[i];

  solve;

#serve per assegnare un valore pari a zero alle
   variabili a e b quando sono diverse da zero ma la
   variabile d[t]=0 per lo stesso nodo t
for {p in FEATURES, t in BRANCH_NODES}{
  if d[t] = 0 then {
    let a[p,t] := 0;
    let a_capp[p,t] := 0;
    let b[t] := 0;
  }
}

for{i in OSSERVAZIONI_VAL}{
   let nodo_foglia_calcolato[i] := 0;
   let classe_calcolata[i] := 0;
}

#questo for serve per calcolare la foglia in cui cade
   un'osservazione x(i)
for{i in OSSERVAZIONI_VAL}{
  let t_val := 1;
  repeat {
```

```
        let temp_val := 0.0;
        for{p in FEATURES}{
            let temp_val := temp_val + a[p, t_val] *
               x_val[i,p];
        }
        if temp_val < b[t_val] then{
            let nodo_foglia_calcolato[i] := t_val*2;
            let t_val := t_val*2;
        }else{
            let nodo_foglia_calcolato[i] := t_val*2 + 1;
            let t_val := t_val*2 + 1;
        }
     }while t_val <= floor(T/2) ;
}

 #questo for serve per calcolare la classe di
    un'osservazione x(i)
 for{i in OSSERVAZIONI_VAL}{
    for{l_val in LEAF_NODES}{
       if c_kt[0, l_val] == 1 then{
          if nodo_foglia_calcolato[i] == l_val then
             let classe_calcolata[i] := 0;
     }
       if c_kt[1, l_val] == 1 then{
          if nodo_foglia_calcolato[i] == l_val then
             let classe_calcolata[i] := 1;
     }
   }
}

 let veri_positivi := 0;
```

```
let veri_negativi := 0;
let falsi_positivi := 0;
let falsi_negativi := 0;
let accuratezza := 0;
let richiamo := 0;
let precisione := 0;


#calcola gli indicatori
for{i in OSSERVAZIONI_VAL}{

   if classe_calcolata[i] == 0 and y_val[i] == 0 then
      let veri_negativi := veri_negativi + 1;

   if classe_calcolata[i] == 1 and y_val[i] == 1 then
      let veri_positivi := veri_positivi + 1;

   if classe_calcolata[i] == 0 and y_val[i] == 1 then
      let falsi_negativi := falsi_negativi + 1;

   if classe_calcolata[i] == 1 and y_val[i] == 0 then
      let falsi_positivi := falsi_positivi + 1;
}

if (veri_positivi + veri_negativi + falsi_positivi +
   falsi_negativi) != 0 then
   let accuratezza := (veri_positivi
      +veri_negativi)/(veri_positivi + veri_negativi +
      falsi_positivi + falsi_negativi);

if (veri_positivi + falsi_negativi) != 0 then
```

```
    let richiamo := veri_positivi / (veri_positivi +
       falsi_negativi);

 if (veri_positivi + falsi_positivi) != 0 then
    let precisione := veri_positivi / (veri_positivi +
       falsi_positivi);

 let sol[1,k] := accuratezza;
 let sol[2,k] := richiamo;
 let sol[3,k] := precisione;

 print 'accuratezza out sample';
 display accuratezza;

 for{i in OSSERVAZIONI}{
    let nodo_foglia_calcolato_in_sample[i] := 0;
    let classe_calcolata_in_sample[i] := 0;
}

 #questo for serve per calcolare la foglia in cui cade
    un'osservazione x(i)
 for{i in OSSERVAZIONI}{
    let t_val := 1;
    repeat {
       let temp_val := 0.0;
       for{p in FEATURES}{
          let temp_val := temp_val + a[p, t_val] *
             x[i,p];
        }

       if temp_val < b[t_val] then{
```

```
            let nodo_foglia_calcolato_in_sample[i] :=
               t_val*2;
            let t_val := t_val*2;
         else
            let nodo_foglia_calcolato_in_sample[i] :=
               t_val*2 + 1;
            let t_val := t_val*2 + 1;
          }
      }while t_val <= floor(T/2) ;
}

 #questo for serve per calcolare la classe di
    un'osservazione x(i)
 for{i in OSSERVAZIONI}{
    for{l_val in LEAF_NODES}{
       if c_kt[0, l_val] == 1 then{
          if nodo_foglia_calcolato_in_sample[i] == l_val
             then
             let classe_calcolata_in_sample[i] := 0;
     }
       if c_kt[1, l_val] == 1 then{
          if nodo_foglia_calcolato_in_sample[i] == l_val
             then
             let classe_calcolata_in_sample[i] := 1;
     }
   }
}

 let veri_positivi := 0;
 let veri_negativi := 0;
 let falsi_positivi := 0;
```

```
let falsi_negativi := 0;
let accuratezza := 0;
let richiamo := 0;
let precisione := 0;

#calcola gli indicatori
for{i in OSSERVAZIONI}{

   if classe_calcolata_in_sample[i] == 0 and y[i] == 0
      then
      let veri_negativi := veri_negativi + 1;

   if classe_calcolata_in_sample[i] == 1 and y[i] == 1
      then
      let veri_positivi := veri_positivi + 1;

   if classe_calcolata_in_sample[i] == 0 and y[i] == 1
      then
      let falsi_negativi := falsi_negativi + 1;

   if classe_calcolata_in_sample[i] == 1 and y[i] == 0
      then
      let falsi_positivi := falsi_positivi + 1;
}

if (veri_positivi + veri_negativi + falsi_positivi +
   falsi_negativi) != 0 then
   let accuratezza := (veri_positivi
      +veri_negativi)/(veri_positivi + veri_negativi +
      falsi_positivi + falsi_negativi);
```

```
if (veri_positivi + falsi_negativi) != 0 then
   let richiamo := veri_positivi / (veri_positivi +
      falsi_negativi);

if (veri_positivi + falsi_positivi) != 0 then
   let precisione := veri_positivi / (veri_positivi +
      falsi_positivi);

let solInSample[1,k] := accuratezza;
let solInSample[2,k] := richiamo;
let solInSample[3,k] := precisione;

print 'accuratezza in sample';
display accuratezza;

print 'altre variabili:'
display a;
display b;
display d;
display c_kt;
display L_t;
display nodo_foglia_calcolato;
display classe_calcolata;

print "fine iterazione" & k;

reset data z;
reset data l;
reset data d;
reset data b;
reset data a;
```

```
   reset data N_kt;
   reset data N_t;
   reset data c_kt;
   reset data L_t;
   reset data x;
   reset data y;
   reset data x_val;
   reset data y_val;
}

   let accuratezza_media := 0;
      for{j in 0..49}{
         let accuratezza_media := accuratezza_media +
            sol[1,j];
}
   let accuratezza_media := accuratezza_media / 50;

   let accuratezza_media_in_sample := 0;
   for{j in 0..49}{
      let accuratezza_media_in_sample :=
         accuratezza_media_in_sample + solInSample[1,j];
}

   let accuratezza_media_in_sample :=
      accuratezza_media_in_sample / 50;
```

File *data.dat:*

```
let N := 147;
let P := 15;
let mu := 0.005;
let alpha := 0.02;
let D := 2;
let N_min := 5;
let T := 2^(D+1) -1;

let OSSERVAZIONI := 1 .. N;
let FEATURES := {1 .. P};
let BRANCH_NODES := 1..floor(T/2);
let LEAF_NODES := floor(T/2) +1 .. T;
let TOTAL_NODES := BRANCH_NODES union LEAF_NODES;
let CLASSI := {0,1};

#inizilizzazione degli ancestor
for{t in TOTAL_NODES}{
let A_L[t] := {} ;
let A_R[t] := {} ;
}

#calcolo degli ancestor
for{t in TOTAL_NODES diff {1} }{
let temp := t;
  repeat {
    if temp mod 2 == 0 then
      let A_L[t] := A_L[t] union {temp/2};
      else
      let A_R[t] := A_R[t] union {floor(temp/2)};
```

```
    let temp := floor(temp/2);
    } while temp/2 >=1 ;
}

#calcolo a_l_max
for{t in LEAF_NODES diff {T}: t mod 2 != 0}{
  let a_l_max[t] := max{m in A_L[t]} m;
}
```

File *model.mod:*

```
#insiemi del modello
set OSSERVAZIONI;
set FEATURES;
set BRANCH_NODES;
set LEAF_NODES ;
set TOTAL_NODES;
set CLASSI;
set A_L {TOTAL_NODES};
set A_R {TOTAL_NODES};

#parametri del modello
param N;
param P;
param T;
param D;
param N_min;
param mu;
param alpha;
param temp;
param Y {i in OSSERVAZIONI, k in CLASSI} = if y[i] == k
   then 1 else -1;
param a_l_max {t in LEAF_NODES diff {15}: t mod 2 != 0};


#variabili del modello
var z {OSSERVAZIONI, TOTAL_NODES} binary ;
var l {LEAF_NODES} binary;
var s {FEATURES, BRANCH_NODES} binary;
var d {BRANCH_NODES} binary;
```

```
var b {BRANCH_NODES};
var a {FEATURES, BRANCH_NODES};
var a_capp {FEATURES, BRANCH_NODES};
var bin {FEATURES, BRANCH_NODES} binary;
var N_kt {k in CLASSI,t in LEAF_NODES} = 1/2 * sum {i in
   OSSERVAZIONI}(1 + Y[i,k])*z[i,t];
var N_t {t in LEAF_NODES} = sum {i in OSSERVAZIONI} z[i,t];
var c_kt {k in CLASSI, t in LEAF_NODES} binary;
var L_t{t in LEAF_NODES} >=0;

/*variabili della parte robusta*/
var eta {OSSERVAZIONI, BRANCH_NODES};
var beta {OSSERVAZIONI, FEATURES, BRANCH_NODES} >= 0;
var pi_greco {OSSERVAZIONI, FEATURES, BRANCH_NODES, 1..5}
   >= 0;


#FUNZIONE OBIETTIVO
minimize obiettivo:
   (1/L_tilda) * (sum {t in LEAF_NODES} L_t[t] )
     + alpha * (sum{t in BRANCH_NODES, j in FEATURES}
        s[j,t] );

#vincloli del modello
subject to eq1 {t in BRANCH_NODES diff {1}}:
  d[t] <= d[floor(t/2)];

subject to eq2_1 {t in BRANCH_NODES}: d[t] == 0 ==> b[t] =
   0;

subject to eq3_1 {t in BRANCH_NODES}:
```

```
  sum{j in FEATURES} s[j,t] >= d[t];

subject to eq3_2 {j in FEATURES, t in BRANCH_NODES}:
  s[j,t] <= d[t];

subject to eq4_1 {j in FEATURES, t in BRANCH_NODES}:
   s[j,t] == 0 ==> a[j,t] = 0;

subject to eq5_1 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] >= a[j,t];

subject to eq5_2 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] >= -a[j,t];

subject to eq5_3 {j in FEATURES, t in BRANCH_NODES}:
  a_capp[j,t] <= -a[j,t] or a_capp[j,t] <= a[j,t];

subject to eq5_5 {t in BRANCH_NODES}: d[t] == 0 ==> sum {j
   in FEATURES} a_capp[j,t] = 0;

subject to eq6_1{t in LEAF_NODES}:
  sum {i in OSSERVAZIONI} z[i,t] >= N_min*l[t];

subject to eq6_2{i in OSSERVAZIONI, t in LEAF_NODES}:
  z[i,t] <= l[t];

subject to eq7 {i in OSSERVAZIONI}:
  sum {t in LEAF_NODES} z[i,t] = 1;

subject to eq8_1 {i in OSSERVAZIONI, t in LEAF_NODES, a_l
   in A_L[t]}:
```

```
  if z[i,t] == 1 then sum {j in FEATURES}
     (a[j,a_l]*x[i,j]) + mu <= b[a_l];

subject to eq8_3 {i in OSSERVAZIONI, t in LEAF_NODES, a_l
   in A_L[t]}:
  z[i,t] == 1 ==> eta[i,a_l] + sum{jj in FEATURES}
     (beta[i,jj,a_l]*( y[i]*varrho_classe1[jj] +
     (1-y[i])*varrho_classe0[jj] ) ) + mu <= b[a_l];

subject to eq8_3_1_DRO {i in OSSERVAZIONI, t in
   LEAF_NODES, tt in A_L[t]}:
  z[i,t] == 1 ==> sum{k in 1..5}( sum{jj in FEATURES}
     h_i[i,jj,k]*pi_greco[i,jj,tt,k] ) <= eta[i,tt];

subject to eq8_3_2_DRO {i in OSSERVAZIONI, j in FEATURES,
   t in LEAF_NODES, tt in A_L[t]}:
  z[i,t] == 1 ==> sum{k in 1..5}( sum{jj in FEATURES}
     (y[i]*C_x_classe1[j,jj,k] +
     (1-y[i])*C_x_classe0[j,jj,k] )*pi_greco[i,jj,tt,k] )
     >= a[j,tt];

subject to eq8_3_3_DRO {i in OSSERVAZIONI, j in FEATURES,
   t in LEAF_NODES, tt in A_L[t]}:
  z[i,t] == 1 ==> sum{k in 1..5}( sum{jj in FEATURES}
     D_x[j,jj,k]*pi_greco[i,jj,tt,k] ) >= -beta[i,j,tt];

subject to eq8_2 {i in OSSERVAZIONI, t in LEAF_NODES, a_r
   in A_R[t]}:
  if z[i,t] == 1 then sum {j in FEATURES}
     (a[j,a_r]*x[i,j]) >= b[a_r];
```

```
subject to eq8_4 {i in OSSERVAZIONI, t in LEAF_NODES, a_r
   in A_R[t]}:
  z[i,t] == 1 ==> -eta[i,a_r] -sum{jj in FEATURES}
     (beta[i,jj,a_r]*( y[i]*varrho_classe1[jj] +
     (1-y[i])*varrho_classe0[jj] ) ) >= b[a_r];

subject to eq8_4_1_DRO {i in OSSERVAZIONI, t in
   LEAF_NODES, tt in A_R[t]}:
  z[i,t] == 1 ==> sum{k in 1..5}( sum{jj in FEATURES}
     h_i[i,jj,k]*pi_greco[i,jj,tt,k] ) <= eta[i,tt];

subject to eq8_4_2_DRO {i in OSSERVAZIONI, j in FEATURES,
   t in LEAF_NODES, tt in A_R[t]}:
  z[i,t] == 1 ==> sum{k in 1..5}( sum{jj in FEATURES}
     (y[i]*C_x_classe1[j,jj,k] +
     (1-y[i])*C_x_classe0[j,jj,k] )*pi_greco[i,jj,tt,k] )
     >= -a[j,tt];

subject to eq8_4_3_DRO {i in OSSERVAZIONI, j in FEATURES,
   t in LEAF_NODES, tt in A_R[t]}:
  z[i,t] == 1 ==> sum{k in 1..5}( sum{jj in FEATURES}
     D_x[j,jj,k]*pi_greco[i,jj,tt,k] ) >= -beta[i,j,tt];

subject to eq9 {t in LEAF_NODES}:
  sum {k in CLASSI} c_kt[k,t] = l[t];

subject to eq10 {k in CLASSI, t in LEAF_NODES}:
  L_t[t] <= N_t[t] - N_kt[k,t] + N*c_kt[k,t];

subject to eq11 {k in CLASSI, t in LEAF_NODES}:
  L_t[t] >= N_t[t] - N_kt[k,t] - N*(1 - c_kt[k,t]);
```

```
subject to eq12_1 {t in LEAF_NODES: t mod 2 == 0}:
  l[t] = d[t/2];

subject to eq12_2 {t in LEAF_NODES diff {T} : t mod 2 !=
   0}:
  l[t] = d[a_l_max[t]];
```

Bibliografia

[1] Altervista. *Grafico k-fold cross validation.* https://pulplearning.altervista.org/cross-validation-cose-e-come-usarla/. 2012.

[2] Dunn J. Bertsimas D. "Optimal classification trees". In: *Machine Learning* (2017), pp. 1039–1082.

[3] Health Emergency Dashboard. *Situazione Covid-19.* https://covid19.who.int. 3/2023.

[4] Daniel Faccini, Francesca Maggioni e Florian A. Potra. "Robust and Distributionally Robust Optimization Models for Linear Support Vector Machine". In: *Computers Operations Research* 147 (2022), p. 105930. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2022.105930. URL: https://www.sciencedirect.com/science/article/pii/S0305054822001861.

[5] Robert Fourer, David M Gay e Brian W Kernighan. *Ampl.* Boyd & Fraser Danvers, MA, 1995.

[6] Maggioni F. Gheza F. Manelli F. Bonetti G. "Un algoritmo di apprendimento automatico per l'ottimizzazione delle cure di pazienti covid-19". In: *Atti dell'Ateneo di Scienze Lettere e Arti di Bergamo.* (2021), pp. 211–222.

[7] Bram L. Gorissen, İhsan Yanıkoğlu e Dick den Hertog. "A practical guide to robust optimization". In: *Omega* 53 (2015), pp. 124–137. ISSN: 0305-0483. DOI: https:

//doi.org/10.1016/j.omega.2014.12.006. URL: https://www.sciencedirect.com/science/article/pii/S0305048314001698.

[8] M. Lichman. *UCI machine learning repository*. http://archive.ics.uci.edu/ml. 2012.

[9] Aringhieri R. Maggioni F. Lanzarone E. Reuter-Oppermann M. Righini G. Vespucci M.T. "Operations Research for Health Care in Red Zone". In: *AIRO Springer Series* 10 (2022), pp. 35–45. ISSN: 2523-7055.

[10] Francesca Maggioni et al. "Machine Learning Based Classification Models for COVID-19 Patients". In: *Annual Meeting of Operational Research Applied to Health Services*. Springer. 2022, pp. 35–46.

[11] Direzione Generale della Prevenzione sanitaria in collaborazione con Istituto Superiore di Sanità. *Covid-19*. https://www.salute.gov.it/portale/nuovocoronavirus/dettaglioFaqNuovoCoronavirus.jsp?lingua=italiano&id=257. 3/2023.