Alma Mater Studiorum · Università di Bologna

Dipartimento di Fisica e Astronomia
Corso di Laurea Magistrale in Astrofisica e Cosmologia

Uso del Machine Learning per l'identificazione di galassie Lyman-Break vicino a quasar luminosi ad alto redshift ($z \gtrsim 6$)

Tesi di Laurea Magistrale

Laureanda
Fabiana Favale

Relatore
Prof. Cristian Vignali

Correlatori
Dr. Fabio Vito
Dr. Marco Mignoli

V Sessione
Anno Accademico 2023/2024

Indice

Abstract

Lo studio delle galassie ad alto redshift ($z \geq 6$), corrispondente a circa 1 Gyr dopo il Big Bang, è fondamentale per comprendere le fasi primordiali dell'Universo, la formazione delle prime strutture cosmiche e la crescita dei buchi neri supermassicci (SMBHs).

Nel corso degli ultimi vent'anni numerose osservazioni hanno portato alla scoperta di quasar ad alto redshift alimentati da SMBHs con masse estremamente elevate ($M_{BH} \sim 10^6 - 10^9 M\odot$), nonostante la giovane età dell'Universo. Secondo la teoria sulla formazione di quasar, questi dovrebbero trovarsi in regioni sovradense (e.g. Overzier et al. 2009; Romano-Diaz et al. 2011), ma le osservazioni hanno fornito risultati contrastanti (Simpson et al. 2014; Willott et al. 2005; Balmaverde et al. 2017). L'identificazione delle Lyman Break Galaxies (LBGs) nei campi attorno ($\sim$ 10Mpc) a quasar ad alto redshift risulta particolarmente complessa a causa delle loro magnitudini molto deboli e della presenza di sorgenti a redshift più basso, come le stelle nane della nostra galassia, che ne condividono in parte le caratteristiche fotometriche.

Il mio lavoro di tesi propone l'uso di tecniche di machine learning, quali Random Forest (RF), Linear Discriminant Analysis (LDA) e Probabilistic Random Forest (PRF), per migliorare l'identificazione delle LBGs a $z > 6$, sfruttando i dati fotometrici in bande ottiche ed infrarosse. Il RF è un modello che crea una "foresta" di alberi decisionali, ciascuno dei quali fornisce una previsione e il risultato finale viene determinato dalla media delle previsioni degli alberi, migliorando così l'accuratezza. L'LDA è una tecnica statistica che cerca di trovare una combinazione lineare delle caratteristiche per separare al meglio le diverse classi, migliorando la discriminazione tra galassie e contaminanti. Il PRF, infine, è una variante del modello RF che tiene conto delle incertezze nei dati, integrandole nel processo di classificazione, e quindi migliorando la robustezza e l'affidabilità delle previsioni. I modelli sviluppati tengono conto delle incertezze osservative per migliorare la separazione nello spazio dei colori ottici/IR tra galassie ad alto redshift e contaminanti. In particolare, per il RF e l'LDA, le incertezze sono state integrate attraverso il metodo Monte Carlo, mentre nel PRF esse sono incluse direttamente nel modello.
Per l'addestramento e la valutazione dei modelli sono stati utilizzati i dati del catalogo SHELLQs (Matsuoka et al. 2016, 2017, 2018, 2019, 2020, 2022) che comprende 38 galassie a redshift

$z \sim 6$ e 96 stelle nane, mentre il set di test è stato costruito utilizzando il catalogo prodotto da Balmaverde et al. (2017) relativo al campo attorno al quasar SDSSJ1030+0524 ($z = 6.28$), comprendente 16 LBGs a $z \sim 6$ e 16 stelle nane. Questi modelli sono stati valutati attraverso varie metriche, come accuracy (la percentuale di previsioni corrette sul totale delle previsioni effettuate), precision (la proporzione di veri positivi tra le previsioni positive effettuate), recall (la proporzione di veri positivi correttamente identificati dal modello) e F1-score (la media armonica tra precision e recall, che bilancia entrambe le metriche).
La combinazione del modello PRF con la tecnica Synthetic Minority Over-sampling Technique (SMOTE), utilizzata per bilanciare il dataset, ha determinato un notevole miglioramento delle performance rispetto agli altri modelli. In particolare, rispetto a RF + SMOTE (che ha prodotto un'accuratezza di circa il 88% nel test test), il PRF ha raggiunto un'accuratezza e un F1-score pari al 97% nel test set. Come applicazione finale della tesi, vari modelli di machine learning, tra quelli implementati, sono stati applicati ad un catalogo di oggetti pre-selezionati come galassie candidate a $z \sim 6$ tramite criteri di colore nel campo del quasar J005006.67+344521.6 a $z = 6.246$, facente parte del programma strategico SQUEEzE, un'iniziativa del telescopio LBT/LBC che effettua imaging multi-banda nell'ottico e nel vicino infrarosso su 15 campi centrati su quasar a $z \sim 6$. I risultati hanno mostrato che sei dei sette oggetti selezionati come candidati a $z \sim 6$ sono stati classificati come galassie, confermando l'affidabilità del modello. La ricerca sviluppata nella mia tesi può fornire un contributo importante nella classificazione delle galassie Lyman-break a $z \gtrsim 6$. L'uso delle tecniche di machine learning per la classificazione di oggetti ad alto redshift si è rivelato uno strumento particolarmente efficace per affrontare le sfide poste dai dati fotometrici, che altrimenti richiederebbero costosi follow-up spettroscopici. In particolare, con l'avvento delle future survey su larga scala come la Legacy Survey of Space and Time (LSST), che genereranno enormi quantità di dati, il machine learning diventa lo strumento principale per l'analisi e la selezione degli oggetti, consentendo di trattare e classificare rapidamente campioni molto più ampi di quelli attualmente disponibili. I risultati attuali suggeriscono che l'inclusione di nuovi dati e l'esplorazione di ulteriori caratteristiche fotometriche, come l'estensione spaziale degli oggetti, potranno ulteriormente affinare i modelli. L'applicazione di questi modelli nel progetto SQUEEzE aiuterà a comprendere meglio la relazione tra SMBHs e galassie circostanti.

Capitolo 1

Introduzione

Lo studio delle galassie ad alto redshift ($z \geq 6$, che corrisponde a $\simeq 1$ Gyr dopo il Big Bang) rappresenta un campo di ricerca di grande rilevanza in astrofisica, poiché consente di esplorare le fasi primordiali dell'Universo e di comprendere i processi fisici che hanno portato alla formazione delle prime strutture cosmiche, alla formazione e alla crescita dei buchi neri supermassicci (SMBHs). Tuttavia, lo studio di queste galassie risulta complicato per via di numerosi fattori, tra cui l'estrema debolezza luminosa delle galassie ad altissimo redshift e la presenza di sorgenti astrofisiche a redshift più bassi che possono simulare le caratteristiche fotometriche delle galassie ad alto redshift usate per identificarle.

In questo capitolo vengono presentati il contesto astrofisico, le sfide osservative legate alla ricerca di galassie ad alto redshift tramite tecniche che usano i colori in bande ottiche e IR, e gli obiettivi di questa tesi, che si propone di affrontare tali problematiche mediante l'impiego di tecniche di "machine learning" applicate ai dati fotometrici disponibili.

1.1 Contesto Astrofisico

Negli ultimi anni sono stati scoperti numerosi quasar ad alto redshift, spingendo la frontiera della ricerca fino a $z \sim 10$ grazie a survey sempre più profonde e sensibili come HST, Euclid e JWST (Fan et al. (2023)). Lo studio di questi oggetti è fondamentale per comprendere i meccanismi fisici legati alla formazione e crescita dei SMBHs.

Un quasar rappresenta il nucleo attivo di una galassia caratterizzato da un'emissione energetica eccezionalmente elevata (magnitudine assoluta in banda ottica $M \lesssim -23$), alimentata dal processo di accrescimento di materia intorno a un SMBH. L'accrescimento genera un disco, in cui le intense forze gravitazionali e le elevate temperature provocano l'emissione di energia su tutto lo spettro elettromagnetico, dal radio ai raggi X. Grazie alla sua luminosità, il quasar supera quella della galassia ospite ed è spesso osservato ad alti redshift, risultando così uno degli

oggetti più distanti e potenti dell'universo (Urry et al. (1995)).

Lo studio dei quasar ad alto redshift è stato possibile grazie alle survey digitali, ossia campagne osservative che sfruttano rivelatori digitali (come i CCD) per acquisire dati in modo automatizzato e con elevata precisione. Queste tecniche, combinate con metodi di selezione basati sul color drop-out[1] — la sorgente non è rilevata in un filtro per via dell'IGM e viene rivelata in un filtro più rosso "adiacente" (ad esempio, Warren et al. 1987) — hanno facilitato l'identificazione di questi oggetti estremamente potenti e lontani. Nei primi anni 2000, la Sloan Digital Sky Survey (SDSS)[2] ha segnato una svolta, portando alla scoperta dei primi quasar a redshift $z > 5$ (Fan et al. 1999) e $z > 6$ (Fan et al. 2001). Successivamente, le survey nel vicino infrarosso (NIR), come UKIDSS (Lawrence et al. 2007) hanno permesso di individuare i primi quasar con $z > 7$ (Mortlock et al. 2011).

Grazie alla continua analisi dei dati e alle osservazioni spettroscopiche di follow-up, vengono scoperti campioni sempre più ampi di quasar ad alto redshift. Attualmente, sono noti circa 1000 quasar con $z > 5$ e oltre 200 con $z > 6$ (Fan et al. (2023)), un numero in continua crescita con le nuove survey come DESI[3] e la futura LSST[4].
Questi quasar risultano essere alimentati da SMBHs che, nonostante l'Universo fosse ancora molto giovane, hanno già raggiunto masse estremamente elevate ($\sim 4 \times 10^7 - 10^{10}$ M$\odot$) (Fan et al. (2023)). I progressi nella ricerca hanno permesso di ottenere informazioni fondamentali sulla loro crescita nell'Universo primordiale.
Questi oggetti mostrano tassi di accrescimento molto elevati ($\dot{M} \sim 2.2 M\odot$/yr), suggerendo che i primi SMBHs crescessero rapidamente, probabilmente in condizioni più favorevoli rispetto a quelle dei quasar a redshift più bassi.
La massa dei SMBHs (10^6-10^9M$\odot$) che alimentano i quasar luminosi osservati a questi redshift fornisce importanti vincoli sui meccanismi di formazione di questi oggetti nell'Universo primordiale, che è ancora un problema aperto.
I principali scenari che vengono considerati per i semi (seed) da cui si formano sono:

- **Light seeds**: derivano dal collasso di stelle massicce di Popolazione III (Pop III), con masse iniziali di circa $10 - 100 M_\odot$. Questi semi devono accrescere massa molto rapidamente per raggiungere le dimensioni degli SMBH osservati a $z \sim 6 - 7$, il che richiede accrescimento continuo ed efficiente (Volonteri (2010)).

- **Intermediate seeds**: si formano tramite il merge di stelle in ammassi densi nelle prime fasi dell'evoluzione cosmica (tipicamente a $z \sim 10-15$). Le interazioni frequenti in ambienti

[1] si veda la sezione 1.3
[2] L'SDSS è una survey comprendente dati fotometrici e spettroscopici.
[3] DESI è una survey comprendente dati fotometrici e spettroscopici.
[4] Legacy Survey of Space and Time (LSST) è una survey comprendente dati fotometrici e spettroscopici.

ad alta densità portano alla coalescenza di stelle massicce, generando buchi neri di massa intermedia, dell'ordine di $\sim 10^3 M\odot$. La formazione degli intermediate seeds dipende fortemente dalle condizioni ambientali nel ammassi e fornisce una via complementare per la rapida crescita dei SMBHs osservati a redshift elevati (Devecchi et al. (2009)).

- **Heavy seeds**: si formano attraverso il collasso diretto di nubi di gas metal-free in aloni di materia oscura massivi ($\sim 10^7 M_\odot$), senza passare attraverso una fase di formazione stellare. Questi buchi neri nascono con masse più elevate, tipicamente tra $10^4 - 10^6 M_\odot$, riducendo la necessità di accrescimento estremo per spiegare gli SMBH ad alto redshift (Begelman et al. (2006)).

Per far crescere buchi neri con una massa $M_{BH} > 10^9 M\odot$ in meno di un miliardo di anni dopo il Big Bang, l'accrescimento di massa sui buchi neri primordiali di piccola massa deve essere stato molto rapido (Volonteri & Rees, 2005). In effetti, per qualsiasi buco nero originato dal collasso di una stella, il tasso di accrescimento richiesto dovrebbe superare il limite di Eddington. Questo limite rappresenta il punto in cui la forza radiale prodotta dalla pressione di radiazione è pari alla forza gravitazionale che attira la materia. In linea di principio, ciò implica che esiste una luminosità massima che un oggetto di massa M può emettere. Assumendo accrescimento sferico, la luminosità di Eddington è:

$$L = 1.38 \times 10^{38} \frac{M}{M\odot} \text{erg s}^{-1} \tag{1.1}$$

Tuttavia, in condizioni particolari (ad esempio, accrescimento tramite flussi di gas o geometrie che riducono l'effetto della radiazione emergente), il gas può continuare ad alimentare il buco nero a velocità superiori a questo limite, permettendogli di crescere molto più rapidamente di quanto previsto dai modelli standard.

Le simulazioni idrodinamiche cosmologiche suggeriscono che i SMBHs osservati a $z \sim 6$ possano essersi formati da semi con masse iniziali comprese tra 10^5 e $10^6 M_\odot$, situati in aloni di materia oscura con masse tra 10^9 e $10^{11} M_\odot$ (Costa et al. 2014). La loro crescita avviene principalmente attraverso episodi intermittenti di accrescimento al limite di Eddington, regolati dal feedback dell'AGN, che alternano fasi di alimentazione intensa a periodi di crescita più moderata. Dai risultati emerge che i buchi neri più massicci si formano in regioni particolarmente dense, situate nei nodi della rete cosmica dove si incontrano filamenti di materia. Questa posizione strategica garantisce un afflusso costante di gas, favorendo una crescita rapida ed efficiente. Al contrario, in ambienti meno densi, la disponibilità di materiale è più limitata e i buchi neri tendono a raggiungere masse più contenute, di poche volte $10^6 M_\odot$ (Figura 1.1).

Sebbene, la teoria preveda la presenza di regioni sovradense attorno ai quasar ($\sim$ 10Mpc) ad alto redshift (Overzier et al. 2009; Romano-Diaz et al. 2011), le osservazioni hanno prodotto risultati contrastanti. Alcuni studi hanno riportato un eccesso di galassie nei dintorni dei quasar

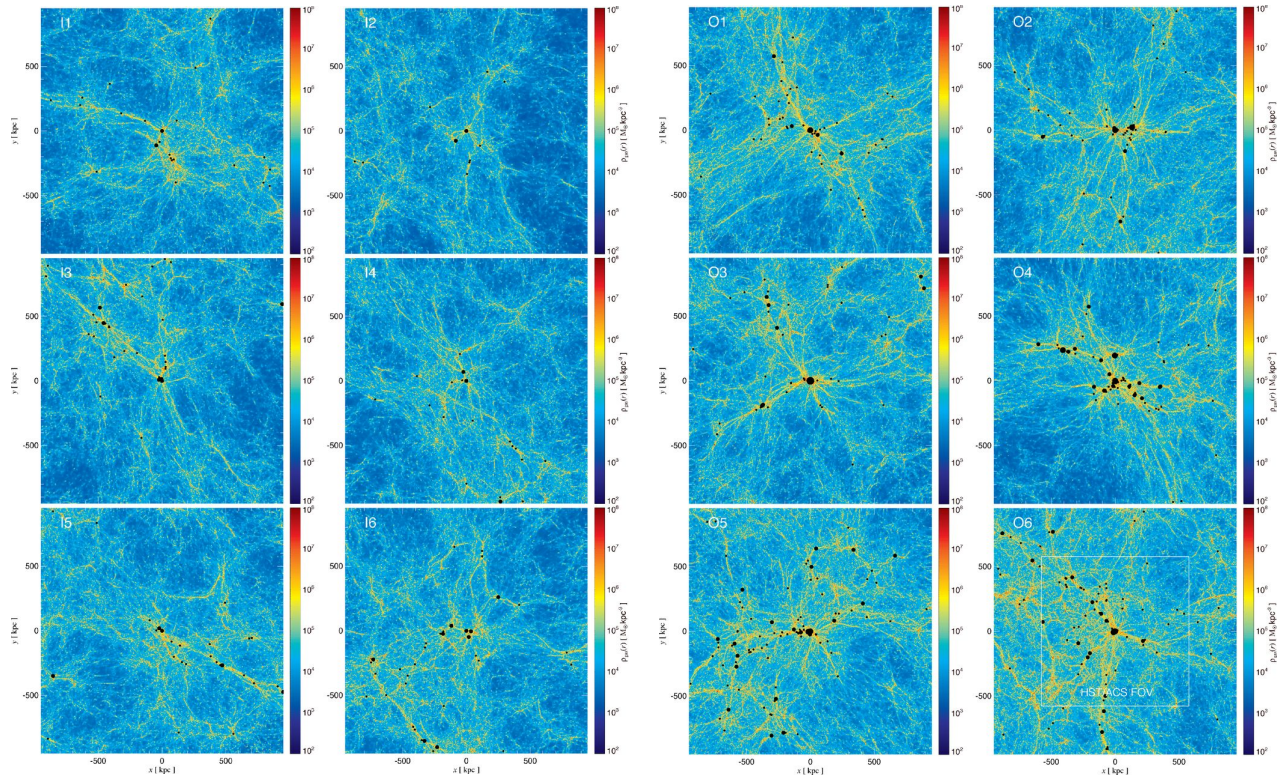


Figura 1.1: Mappe della densità del gas pesate per la massa, proiettate in una "slice" di spessore ≈ 1.9Mpc a $z = 6.2$. (a) Regioni con sovradensità intermedia: la massa tipica dei buchi neri è di qualche decina di milioni di masse solari ($\sim 10^7 M\odot$). (b) Regioni sovradense: i buchi neri più massicci si formano in aloni con M$\approx 4 \times 10^{12} M\odot$, situati all'intersezione di filamenti molto prominenti. Il quadrato bianco nella regione O6 indica il campo di vista di HST/ACS. In tutte le mappe, i punti neri rappresentano la posizione delle "particelle di buco nero" all'interno del cubo proiettato. Queste particelle non simulano in dettaglio la fisica interna dei buchi neri, ma fungono da rappresentazioni numeriche semplificate (punti di massa) che incarnano le proprietà chiave, come la massa e il tasso di accrescimento. La dimensione di ciascun punto è proporzionale alla massa del buco nero corrispondente, consentendo di visualizzare chiaramente le variazioni nell'abbondanza e nella distribuzione dei buchi neri, fenomeno influenzato anche dalla varianza cosmica. L'immagine è presa da Costa et al. (2014)

a $z \gtrsim 5$ (Kashikawa et al. 2007; Utsumi et al. 2010), per esempio in Balmaverde et al. 2017 è stata stimata una sovradensità $\delta \sim 2.4$ [5] con una significatività superiore a 4σ, mentre altri non hanno trovato evidenze significative di sovradensità (Willott et al. 2005; Bañados et al. 2013; Mazzucchelli et al. 2017). In alcuni casi, sono stati persino riportati ambienti sotto-densi (Simpson et al. 2014).

[5] $\delta = \frac{\rho - \overline{\rho}}{\overline{\rho}}$, dove ρ è la densità locale e $\overline{\rho}$ è la densità media della regione o dell'universo considerato. Un valore positivo di($\delta > 0$) indica una sovradensità (cioè una regione in cui la densità è superiore alla media), mentre un valore negativo indica una sottodensità.

Con il lancio del James Webb Space Telescope (JWST), grazie alle sue capacità di spettroscopia slitless ad ampio campo del modulo NIRCam, sono state trovate sovradensità di emettitori di [O III] ($\lambda_{[OIII]}$5007Å) nei campi di quasar massicci osservati, nonostante il campo di vista limitato (due rivelatori da 2.2×2.2 arcmin ciascuno).

Ad esempio, Kashino et al. (2023) hanno studiato un'area di 6.5×3.4 arcmin attorno a J0100+2802 e hanno trovato 24 sistemi emettitori di [O III] esattamente al redshift del quasar, un numero significativamente maggiore rispetto a quelli a redshift inferiori, corrispondendo a un incremento di almeno un fattore 2 (o superiore) rispetto alla densità tipica. Inoltre, Wang et al. (2023) hanno scoperto una struttura filamentare attorno al quasar J0305-3150 a $z = 6.6$, che presenta 10 oggetti con emissione nella riga dell'[O III], con una sovradensità pari a $\delta = 12.6$ in un singolo puntamento di NIRCam.

Questi risultati dimostrano il potenziale di JWST nel rivelare le componenti più deboli delle sovradensità galattiche. Tuttavia, un approccio complementare consiste nell'utilizzare strumenti di largo campo, come il Large Binocular Camera (LBC) sul Large Binocular Telescope (LBT) o il Wide-field Infrared Camera (WIRCam) sul Canada–France–Hawaii Telescope (CFHT). Questi strumenti permettono di ottenere un'ampia e profonda copertura nel visibile e nel vicino infrarosso (NIR) nel campo del quasar, con un campo visivo di $\sim 25 \times 23$ arcmin nei filtri r, i, z, Y e J. La prima identificazione spettroscopica delle strutture di alta densità galattica intorno al QSO centrale è avvenuta nel campo del QSO J1030 (Balmaverde et al. 2017, Mignoli et al. 2020) in cui le galassie sono state trovate utilizzando la tecnica della selezione "Lyman Break" (si veda la sezione 1.2).

1.2 Identificazione di "Lyman Break Galaxies" (LBG) ad alto redshift

Negli ultimi due decenni, diversi studi hanno cercato di identificare galassie in formazione stellare attorno ai quasar a $z > 6$, principalmente attraverso la selezione fotometrica di candidate LBGs (Willott et al. 2005; Zheng et al. 2006; Kim et al. 2009; Morselli et al. 2014; Simpson et al. 2014). Le LBGs sono galassie ad alto redshift ($z > 2$) la cui individuazione si basa su un fenomeno fisico chiamato Lyman Break, che è il risultato dell'interazione tra la luce emessa da queste galassie e il gas di idrogeno neutro presente nello spazio intergalattico (IGM).
Le giovani stelle massicce all'interno delle LBGs emettono una grande quantità di luce UV. Tuttavia, questa radiazione viene in parte assorbita dagli atomi di idrogeno neutro presenti lungo la linea di vista. In particolare, i fotoni con lunghezza d'onda inferiore a 912 Å (il cosiddetto limite di Lyman) hanno sufficiente energia per ionizzare l'idrogeno e vengono quindi completamente assorbiti. Questo crea un calo drastico nel flusso della galassia a queste lunghezze d'onda, il cosiddetto Lyman Break.

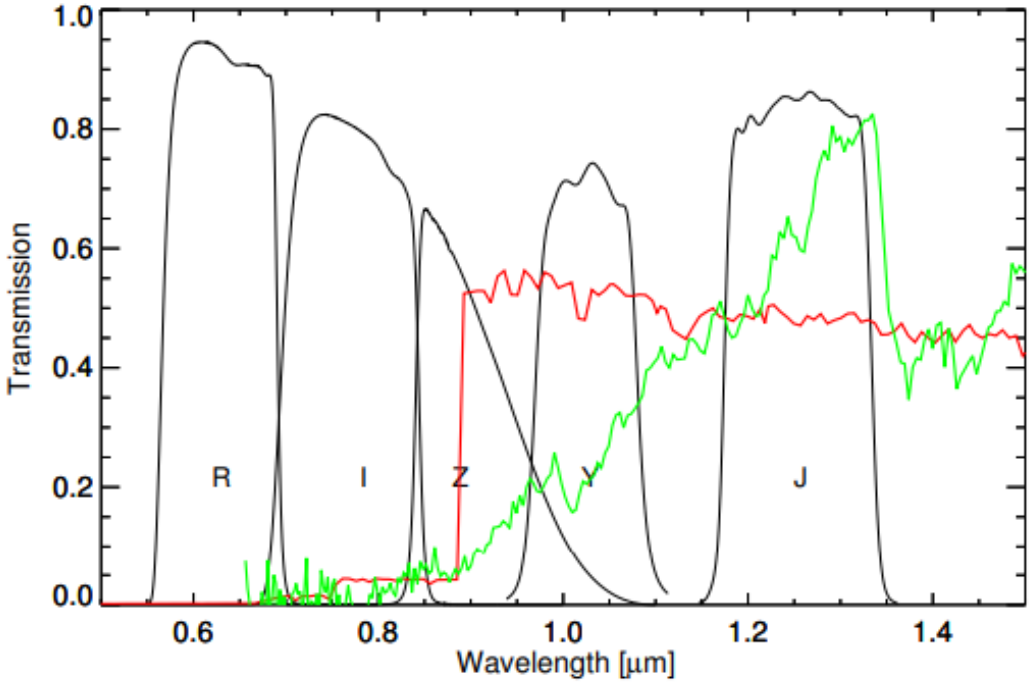


Figura 1.2: risposta dei filtri r, i, z, Y, J e modelli spettrali di una galassia ad un redshift di 6.3 (in rosso) di una brown dwarf (in verde). L'immagine è presa da Balmaverde et al. (2017)

A lunghezze d'onda comprese tra 912Å e 1216Å , invece, si osserva una serie di linee di assorbimento, note come foresta di Lyman-α, dovute alla presenza di nubi di idrogeno a diverse distanze. Questi effetti rendono le LBGs facilmente riconoscibili nel loro spettro di luce. Man mano che la galassia si trova a redshift più elevati, la luce emessa da queste galassie viene spostata verso il rosso a causa dell'effetto Doppler. Questo significa che il Lyman Break e la Lyα forest, che in origine si trovano nell'UV, appaiono a lunghezze d'onda sempre maggiori. Grazie a questa caratteristica, è possibile identificare le LBGs osservando il loro colore in diverse bande fotometriche. In particolare per $z \sim 6$, il break si sposta nella banda i, permettendo la selezione delle cosiddette i-dropout galaxies.

In Figura 1.2 è mostrato in rosso il template di una galassia a $z > 6$. Si può osservare come le galassie ad alto redshift siano ben rilevabili nel filtro z, presentino un forte calo nella banda i, mentre risultino non rivelabili nel filtro r. Questi rappresentano i criteri fondamentali adottati per la selezione iniziale di tali galassie.

Diversi contaminanti a redshift più bassi possono rientrare nella selezione fotometrica, come le galassie starburst polverose, gli AGN oscurati, galassie passive a $z \sim 1.1$ (che presentano il D4000 a 4000Å) che possono imitare il Lyman break tipico delle galassie ad alto redshift (e.g.

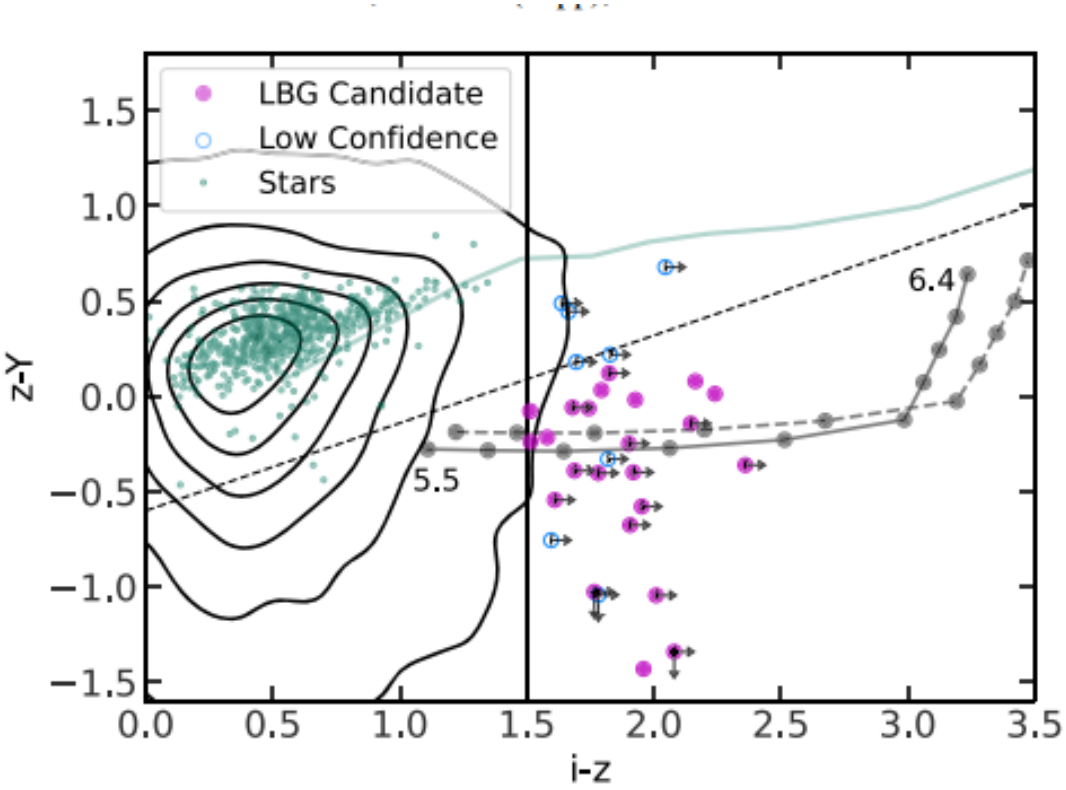


Figura 1.3: Diagramma colore-colore con candidati LBG (punti magenta pieni), candidati a bassa confidenza (cerchi blu aperti) e sorgenti stellari (punti verdi) nel campo. Questo mostra il colore $z - Y$ rispetto al colore $i - z$. La linea verticale mostra il cut del colore a $i - z < 1.5$, mentre la linea tratteggiata diagonale mostra il cut per rimuovere i contaminanti stellari. Le tracce grigie in ogni grafico mostrano i colori teorici di una galassia giovane in formazione stellare a redshift che vanno da $z = 5.5$ a $z = 6.4$, con intervalli di redshift di $\Delta z = 0.1$. I contorni neri mostrano la regione in cui risiedono tutte le sorgenti rilevate nel campo J0100. La traccia verde mostra i colori teorici delle nane brune MLT a metallicità solare calcolati usando la griglia del modello Sonora (Marley et al. 2021). Le frecce sui punti magenta e blu mostrano i limiti inferiori (superiori) sui colori $i - z$ ($z - Y$) a causa di non-rilevamenti nelle bande i e Y. L'immagine è presa da (Pudoka et al. 2024)

Dunlop et al. 2013; Finkelstein et al. 2015). I più numerosi contaminanti sono attesi essere stelle nane di tipo M, L e T nella nostra galassia, a causa dei loro colori molto rossi che possono coincidere con quelli delle LBGs candidate ad alto z e che possono presentare una morfologia puntiforme simile agli oggetti compatti ad altissimo redshift.
Successivamente, nel processo di identificazione di LBGs ad alto redshift vengono applicati criteri di selezione basati sul colore, poiché ci si aspetta che le LBGs occupino una regione specifica nei diagrammi colore-colore che coinvolgono magnitudini nel vicino infrarosso. In particolare, i diagrammi che utilizzano le bande i, z e Y si sono dimostrati efficaci nell'isolare candidate LBG a $z > 6$ (Bowler et al. 2015; Matsuoka et al. 2016). Infatti, sempre in Figura

1.2, è mostrato in verde il template di una nana bruna; queste stelle fredde presentano un colore $(i - z)$ simile a quello delle LBGs a $z \sim 6$. Tuttavia, mentre lo spettro delle BDs continua a salire nell'IR, quello delle galassie risulta più piatto. Dunque, grazie ai dati aggiuntivi forniti dalle bande Y e J, è possibile distinguere la maggior parte di questi oggetti, poiché appaiono significativamente più rossi rispetto alle galassie ad alto redshift nei colori $z-Y$ e $z-J$ e dunque si posizionano in zone differenti dei diagrammi colore-colore rispetto alle LBGs come mostrato in Figura 1.3. Rimane però un certo grado di contaminazione che ha un impatto sull'interpretabilità dei dati e quindi sull'identificazione di strutture di galassie attorno a QSOs ad alto redshift.

1.3 Obiettivi della tesi

In questo contesto si sviluppa il lavoro di questa tesi, che adotta un approccio basato su algoritmi di machine learning per identificare LBGs ad alto redshift nelle vicinanze di quasar a $z > 6$, utilizzando dati fotometrici provenienti da survey osservative. L'obiettivo principale è verificare l'efficacia delle tecniche di classificazione supervisionata nel distinguere le galassie ad alto redshift da altre sorgenti astrofisiche, in particolare dalle brown dwarfs, in osservazioni fotometriche ottiche/IR di campi contenenti QSOs a $z \gtrsim 6$.

In una prima fase, addestreremo diversi modelli di machine learning (ad es. Random Forest) utilizzando set di dati classificati (si veda appendice A), per valutare e ottimizzare la loro capacità di selezionare candidati LBG in modo accurato e gestire in maniera rigorosa le incertezze osservative. Alcuni dei modelli addestrati verranno successivamente applicati per identificare galassie attorno a uno dei quasar coinvolti nelle osservazioni del progetto SQUEEzE, un programma strategico del telescopio LBT/LBC che effettua imaging multi-banda nell'ottico e nel vicino infrarosso su 15 campi di quasar a $z \sim 6$. L'obiettivo in questo contesto è selezionare candidati affidabili di LBG mediante la tecnica dell'*i*-band drop-out, fino a magnitudini di $z \sim 25$, per vincolare l'ambiente su larga scala dei quasar a $z \sim 6$.

La strategia adottata si ispira a precedenti studi condotti sul campo J1030+0524 (Morselli et al. 2014; Balmaverde 2017), che hanno permesso l'identificazione di galassie fino a 8 Mpc, basandosi su criteri fotometrici quali $(i - z) > 1.3$, $(z - Y) < 1$ e l'assenza di rilevazione nella banda r (si veda la sezione 2.3). Pur ispirandoci a questi criteri, in questo lavoro adottiamo una selezione simile ma non identica, modificata per meglio adattarsi alle caratteristiche specifiche del nostro dataset.

L'uso di tecniche di machine learning per la classificazione di oggetti a redshift elevato si rivela uno strumento particolarmente efficace per affrontare le sfide poste dai dati fotometrici, che altrimenti richiederebbero follow-up spettroscopici onerosi per confermare l'identità degli oggetti. Questo approccio consente di integrare in modo rigoroso le incertezze osservative, mi-

gliorare la separazione tra le diverse classi di sorgenti e ottimizzare l'accuratezza nella selezione degli LBGs attorno ai quasar a $z \sim 6$. In definitiva, per campioni di grandi dimensioni, il machine learning rappresenta lo strumento più efficace per la selezione e la classificazione di oggetti particolari, offrendo un metodo scalabile e robusto che potrà essere applicato non solo al campione SQUEEzE, ma anche a futuri dataset provenienti da survey su larga scala.

Alla luce degli obiettivi di questa tesi, la struttura del lavoro è articolata come segue:

- Nel Capitolo 2 vengono descritti i dataset utilizzati per l'analisi, SHELLQs e J1030, evidenziandone le caratteristiche principali e la distribuzione delle classi.
- Nel Capitolo 3 si approfondisce il funzionamento teorico degli algoritmi di classificazione implementati, ovvero Random Forest (RF), Linear Discriminant Analysis (LDA) e Probabilistic Random Forest (PRF).
- Nel Capitolo 4 viene trattata la fase di preprocessing dei dati, includendo la gestione delle incertezze, il bilanciamento del dataset e l'implementazione pratica degli algoritmi di classificazione.
- Infine, nel Capitolo 5, gli algoritmi migliori vengono applicati ai dati fotometrici ottenuti per il campo del quasar J005006.67+344521.6 a $z = 6.25$, analizzando e discutendo i risultati ottenuti. In questa sezione vengono anche tratte le conclusioni, sintetizzando i risultati del lavoro e proponendo possibili sviluppi futuri.

Capitolo 2

Descrizione dei dataset

La comprensione e l'analisi dei dataset utilizzati costituiscono una parte fondamentale di questo lavoro, poiché influenzano direttamente la qualità e l'affidabilità dei modelli sviluppati. In questa sezione vengono presentati i due dataset principali impiegati nell'analisi: il dataset SHELLQs, e il dataset J1030. Viene inoltre descritto il contenuto di ciascun dataset, con particolare attenzione alle classi e alle caratteristiche considerate.

Prima di procedere alla descrizione dei cataloghi è importante comprendere in generale come funziona l'addestramento di un modello.

2.1 Funzionamento del training

Per addestrare un modello e valutarne le prestazioni, è necessario disporre di un set di dati che viene suddiviso, in genere, in tre parti: **training set**, **validation set** e **test set**. Le proporzioni più comuni sono:

- Training set: 70-80% dei dati
- Validation set: 10-15% dei dati
- Test set: 5-15% dei dati

Le fasi del processo sono le seguenti:

1. **Training (addestramento)**: Il **training set** viene utilizzato per addestrare il modello. Questo set contiene sia le caratteristiche degli oggetti che la loro classe di appartenenza. Durante l'addestramento, l'algoritmo analizza questi dati per identificare criteri o regole che permettano di classificare correttamente gli oggetti.
2. **Validation (validazione)**: Una volta completato l'addestramento, il modello viene testato con i dati del **validation set**. In questa fase, il modello applica le regole apprese

in fase di training ai dati di validazione e produce una classificazione. Successivamente, i risultati vengono confrontati con le vere classi di appartenenza per calcolare le metriche di prestazione, come l'accuratezza. Se necessario, si procede con l'ottimizzazione del modello, ad esempio aggiustando i parametri per migliorare le sue prestazioni.

3. **Testing (valutazione finale)**: Infine, si utilizza il **test set** per una valutazione definitiva delle prestazioni del modello. Questo set di dati, non precedentemente "visto" dal modello durante l'addestramento e la validazione, serve a confermare se il modello generalizza bene anche su dati nuovi.

2.2 SHELLQs

Il dataset utilizzato per il training e il validation set proviene dal catalogo SHELLQs[1], un catalogo contenente 313 oggetti con identificazione spettroscopica:

- 162 quasar;
- 38 galassie;
- 17 emettitori di [O III];
- 96 brown dwarfs.

(Matsuoka et al. (2016), (2017), (2018), (2019), (2020), (2022)).

Il progetto SHELLQs ha portato alla scoperta di quasar e galassie brillanti a redshift compresi tra 5.7 e 6.9. Questi risultati provengono da dati raccolti dal telescopio Subaru con lo strumento di imaging a grande campo Hyper Suprime-Cam (HSC) e rappresentano un successo significativo nell'identificazione fotometrica di oggetti a magnitudini relativamente brillanti ($z < 23.5$ mag).

L'obiettivo principale del progetto è studiare le proprietà e la distribuzione di quasar e AGN a bassa luminosità [2] nell'epoca della reionizzazione ($z \approx 6 - 7$), esplorando anche la funzione di luminosità delle galassie e dei quasar ad alto redshift. A tal fine, il progetto utilizza osservazioni fotometriche su aree di cielo molto vaste per aumentare il campione di quasar ad alto redshift utilizzando le bande g, r, i, z, Y e applicando criteri di colore per selezionare oggetti candidati ad essere ad alto redshift. Il catalogo qui utilizzato come training set include oggetti che hanno passato tale pre-selezione in colore e che sono poi stati identificati spettroscopicamente come QSOs a $z \approx 5.7 - 7$, galassie a $z \approx 5.7 - 7$ e nane brune, che sono i contaminanti più comuni.

[1] Subaru High-z Exploration of Low-Luminosity Quasars.

[2] i quasar "SDSS-like" hanno tipicamente una M_{1450} tra -26 e -27, mentre quelli di SHELLQs si estendono in un range di circa -22 a -25.

2.2.1 Osservazioni e analisi dati

Le osservazioni sono state effettuate con l'Hyper Suprime-Cam (HSC), una camera a grande campo (diametro $\sim 1.5°$) montata sul telescopio Subaru da 8.2 m situato sulla montagna Mauna Kea nell'isole Hawaii. Sono attualmente disponibili cinque filtri a banda larga (g, r, i, z e Y) e diversi filtri a banda stretta.

La survey HSC-SSP è suddivisa in tre livelli con diverse combinazioni di area e profondità. La survey Wide punta a osservare 1400 deg^2, principalmente lungo l'equatore celeste, attraverso i cinque filtri a banda larga. I tempi totali di esposizione variano da 10 minuti nelle bande g e r a 20 minuti nelle bande i, z e Y, suddivisi in esposizioni individuali di circa 3 minuti ciascuna. Le magnitudini limite a 5σ sono $(g, r, i, z, Y) = (26.5, 26.1, 25.9, 25.1, 24.4)$ mag.

La survey Deep e Ultra-Deep coprono rispettivamente 27 e 3.5 deg^2 e sono collocate in regioni del cielo coperte da survey note (come XMM-Newton Deep Survey). Utilizzano cinque filtri a banda larga e quattro filtri a banda stretta, con l'obiettivo di raggiungere una profondità limite a 5σ di $r = 27.1$ mag (Deep) o $r = 27.7$ mag (Ultra-Deep).

Il set di filtri utilizzato in questa survey è sensibile ai quasar con redshift fino a $z \approx 7.4$. Tuttavia, la capacità di selezione cala bruscamente a $z > 7$, dove la banda Y è influenzata dal Gunn-Peterson trough, limitando così la survey a oggetti intrinsecamente molto luminosi a questi redshift.

Il progetto SHELLQs beneficia anche di dati NIR di archivio da UKIDSS e VIKING. Lo UKIDSS è una survey di imaging che utilizza WFCAM, una fotocamera a largo campo montata sul telescopio UKIRT da 3.8 m (Lawrence et al. 2007). La Large Area Survey copre la maggior parte dell'area della survey HSC, con magnitudini limite a 5σ di (Y, J, H, K) = (20.9, 20.4, 20.0, 20.1) mag. VIKING è una delle survey pubbliche dell'ESO con il telescopio VISTA da 4.1 m, con l'obiettivo di osservare 1500 deg^2 del cielo e magnitudini limite a 5σ di (Z, Y, J, H, K) = (23.1, 22.3, 22.1, 21.5, 21.2) mag.

2.2.2 Selezione dei quasar

I quasar ad alto redshift sono caratterizzati da colori ottici estremamente rossi nelle bande i e z, causati dal forte assorbimento dell'IGM a lunghezze d'onda più corte rispetto alla riga di Lyα (vedi 1.1). Esistono tre principali fonti di contaminazione astrofisica nella selezione fotometrica dei quasar. La prima è rappresentata dalle brown dwarfs galattiche, a causa dei loro colori molto rossi e della loro apparenza puntiforme. La seconda fonte sono le galassie rosse a $z \sim 1$, il cui "break" a 4000 Å porta a colori $i - z$ rossi; tuttavia, ci si aspetta che l'eccellente qualità delle immagini dell'HSC aiuti a identificare morfologicamente queste galassie a basso redshift come sorgenti estese spazialmente. La terza fonte sono le galassie Lyman-break (LBG) deboli a $z \sim 6$, anch'esse influenzate dall'assorbimento dell'IGM. Per selezionare i candidati quasar per prima

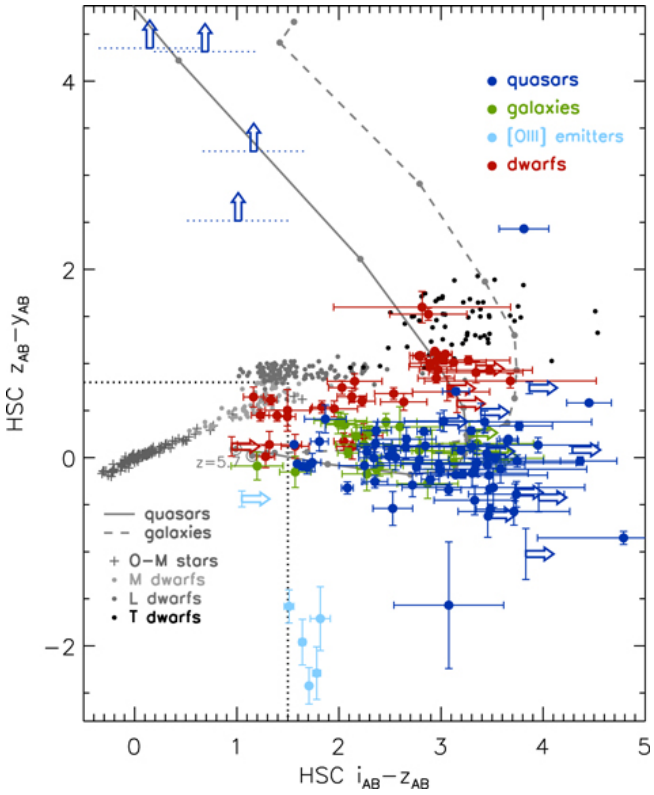


Figura 2.1: I colori HSC ($i_{AB} - z_{AB}$) e ($z_{AB} - y_{AB}$) dei quasar di SHELLQs (punti blu; sono inclusi i quasar precedentemente noti), delle galassie (punti verdi), degli emettitori di [O III] (punti azzurri) e delle nane fredde (punti rossi). Le croci e i punti grigi rappresentano le stelle della Via Lattea (Pickles 1998) e le nane brune, mentre le linee continue e tratteggiate rappresentano i modelli di quasar e galassie a $z \geq 5.7$; i punti lungo le linee indicano i redshift con passi di 0.1, con $z = 5.7$ segnato dai grandi cerchi vuoti. Le linee tratteggiate rappresentano i criteri di colore utilizzati nella nostra ricerca nel database HSC. I quattro quasar nell'angolo in alto a sinistra (indicati dalle frecce verso l'alto) non vengono rilevati nelle bande i e z, e sono riportati in posizioni orizzontali arbitrarie.

cosa sono stati selezionati gli oggetti dal database HSC-SSP che soddisfano i seguenti criteri :

$$(z_{AB} < 24.5 \text{ and } \sigma_z < 0.155 \text{ and } i_{AB} > 1.5 \text{ and } z_{AB} - z_{cModel,AB} < 0.3) \tag{2.1}$$

o

$$(y_{AB} < 24.0 \text{ and } \sigma_y < 0.155 \text{ and } z_{AB} - y_{AB} > 0.8 \text{ and } y_{AB} - y_{cModel,AB} < 0.3) \tag{2.2}$$

dove la magnitudine cModel (m_cModel) viene definita adattando al profilo della sorgente un modello a due componenti, solitamente costituito da:

- un profilo de Vaucouleurs (che rappresenta il bulge)
- un profilo esponenziale (che rappresenta il disco)

entrambi questi componenti sono convoluti con la PSF, e le σ rappresentano le fluttuazioni rispetto al background. Le condizioni dell'equazione 2.1 selezionano i dropout in banda i a $z \sim 6$, mentre quelle dell'equazione 2.2 selezionano i-dropout in banda z a $z \sim 7$. Questi tagli di colore sono utilizzati per escludere stelle relativamente blu con tipi spettrali da O a M, mentre la differenza tra le magnitudini PSF (ottenuta adattando un modello di PSF al profilo della sorgente) e cModel è utilizzata per escludere sorgenti estese. Successivamente sono stati rimossi gli oggetti a basso redshift che presentano un segnale superiore a 3σ nella banda g o r, infine sono rimossi anche gli oggetti il cui profilo non rispecchia una PSF, cioè quelli troppo concentrati, troppo estesi o con elevata ellitticità. I candidati selezionati sopra sono poi stati abbinati ai cataloghi UKIDSS e VIKING entro 1" nell'area di sovrapposizione della survey. La profondità delle survey HSC e VIKING consente di identificare e rimuovere oggetti come le nane brune di tipo L-T, che presentano colori medi di $zAB - J_{AB} \sim 2 - 4$mag.

I campioni restanti sono quindi stati elaborati attraverso un algoritmo probabilistico bayesiano. Questo calcola la probabilità a posteriori che ogni sorgente sia un quasar ad alto redshift anziché una stella o una nana rossa, basandosi sulla fotometria in tutte le bande disponibili, oltre che su modelli di distribuzione spettrale di energia (SED) e di densità superficiale delle popolazioni considerate.

Per una sorgente rilevata con le quantità osservate d, la probabilità bayesiana P_Q^B di essere un quasar è data da:

$$P_Q^B(\mathbf{d}) = \frac{W_Q(\mathbf{d})}{W_Q(\mathbf{d}) + W_D(\mathbf{d})}$$

con

$$W_{Q/D}(\mathbf{d}) = \int S(\mathbf{p}) Pr(det|\mathbf{p}) Pr(\mathbf{d}|\mathbf{p}) d\mathbf{p} \tag{2.3}$$

dove Q e D denotano rispettivamente un quasar o una brown dwarf. Il vettore $\mathbf{d}$ rappresenta le magnitudini in tutte le bande disponibili, mentre $\mathbf{p}$ tutte le proprietà intrinseche della sorgente (cioè luminosità e redshift per un quasar, e luminosità e tipo spettrale per una brown dwarf). Le funzioni $S(p)$, $Pr(det|p)$ e $Pr(d|p)$ rappresentano la densità superficiale, la probabilità che la sorgente sia rilevata e la probabilità che la sorgente abbia le quantità osservate d, rispettivamente, ciascuna in funzione di p. $S(p)$ è calcolata con la funzione di luminosità dei quasar di Willott et al. 2010 e il modello di brown dwarf galattica di J. A. Caballero e Klement 2008. $Pr(det|d)$ è impostata arbitrariamente a 1 per $z_{AB} < 26$ mag o $y_{AB} < 25$ mag e a 0 altrove. A causa del campionamento discreto dei modelli di brown dwarf raggruppati in tipi spettrali individuali, l'integrazione nell'equazione 2.3 è trattata come una somma per i tipi spettrali.

Gli oggetti con probabilità $P_Q^B > 0.1$ vengono aggiunti al campione di candidati. Un sottoinsieme di questi viene selezionato per osservazioni spettroscopiche di follow-up con strumenti

come FOCAS del telescopio Subaru o strumenti del Gran Telescopio Canarias (GTC). Questo passaggio è cruciale per confermare la natura dei candidati. La spettroscopia di follow-up mira a rilevare caratteristiche tipiche dei quasar, come le ampie righe di emissione (ad esempio, Lyα e N V) e drop del continuo indicative del Gunn-Peterson trough. La presenza di queste caratteristiche conferma gli oggetti come quasar ad alto redshift e permette la misurazione precisa del redshift dell'oggetto.

Durante questa fase sono stati trovati spettri relativi a galassie ad alto redshift, brown dwarfs e [OIII] emitters a $z \sim 0.8$. Quelli delle galassie ad alto redshift sono caratterizzati dall'assenza di righe di ionizzazione elevata, ampie righe di emissione o un continuo blu. Alcune galassie ad alto redshift presentano righe strette di Lyα e bruschi cali nel continuo caratteristici del Gunn-Peterson troughs. Sono chiaramente visibili anche le righe di assorbimento interstellare di Si II λ1260, Si II λ1304 e C II λ1335, indicando che si tratta di galassie Lyman Break (LBG). La distribuzione dei redshift di queste galassie risulta essere sistematicamente diversa da quella dei quasar scoperti, in quanto le galassie tendono a concentrarsi in un intervallo di redshift leggermente più basso rispetto ai quasar. Questo è dovuto, da un lato, al fatto che solo gli oggetti intrinsecamente luminosi possono essere rilevati a $z > 6.0$, dove il Gunn-Peterson trough entra nella banda z. Dall'altro, le galassie sono più rosse dei quasar a $6.0 < z < 6.3$, a causa di righe di Lyα più deboli[3] e continuo più rosso, rendendole più difficili da distinguere dalle brown dwarfs galattiche.
Gli spettri delle brown dwarfs presentano un continuo più rosso e le loro classi spettrali sono state determinate confrontando i template standard spettrali delle nane di tipo M4 a T8 con gli spettri osservati nel range $\lambda_{obs} = 7500 - 9800$ Å.

In questo lavoro, sono state selezionate le magnitudini nelle bande i, Y e z insieme alle etichette di classe. L' estensione delle classi prese in considerazione nel dataset sono le seguenti:

- Quasar: 162 oggetti,
- Brown Dwarf: 96 oggetti,
- Galassie: 38 oggetti,

Sono stati esclusi gli emettitori [OIII] nel resto della tesi in quanto le osservazioni a cui verranno applicati i modelli non sono abbastanza profonde da rilevarne in numero significativo.

2.3 J1030

Il dataset utilizzato per il test set proviene dal catalogo costruito da Balmaverde 2017. Il campo intorno al quasar SDSSJ1030+0524 (J1030) a $z = 6.28$ è unico per la sua copertura

[3]L'EW a riposo per i quasar varia attorno valori $\sim 70 - 100$Å per le LBGs varia attorno a valori $\sim 5 - 10$Å

multi-banda e rappresenta un eccellente set di dati per lo studio dell'ambiente attorno a un SMBH primordiale.

2.3.1 Osservazioni e analisi dati

Le osservazioni ottiche sono state effettuate con il Large Binocular Telescope (LBT), un telescopio binoculare con due specchi da 8.4m ciascuno, situato sul Monte Graham, in Arizona, utilizzando la Large Binocular Camera (LBC). Le osservazioni nelle bande Y e J sono invece state effettuate con il Canada-France-Hawaii Telescope (CFHT), un telescopio da 3.6 metri situato a Mauna Kea, Hawaii, utilizzando la Wide-field InfraRed Camera (WIRCam) che ha un campo di vista di $21' \times 21'$, simile a quello di LBT/LBC.

2.3.2 Selezione delle galassie

La maggior parte delle galassie spettroscopicamente confermate a redshift ~ 6 mostra colori $i - z$ maggiori di 1.3 a causa del Gunn-Peterson trough (e.g. E. Vanzella et al. 2009).
Sono stati selezionati oggetti suddivisi in tre gruppi distinti in base ai seguenti criteri di colore:

1. $i - z > 1.3$ e $z < 25.2$
2. $1.1 < i - z < 1.3$ e $z < 25.2$
3. $i - z > 1.3$ e $z > 25.2$

A magnitudini deboli la densità numerica prevista di LBG a $z \sim 6$ è più alta rispetto a quella delle stelle nane, poiché il numero previsto di stelle nane diminuisce a magnitudini più deboli, ovvero per $z > 25.5$, mentre il conteggio atteso di LBG ad alto redshift aumenta. La selezione di LBG ad alto z dovrebbe quindi essere meno suscettibile alla contaminazione stellare sotto questa soglia di magnitudine.

Tutti i candidati sono rilevati nella banda z e mostrano limiti superiori per il colore $z - Y$ se non rilevati nella banda Y, e limiti inferiori per il colore $i - z$ se non rilevati nella banda i.

Per effettuare una classificazione morfologica, è stato utilizzato il diagramma diagnostico mag(apertura) vs. mag(totale), poiché il rapporto tra queste due quantità rappresenta un indice robusto di concentrazione della sorgente. Applicando questo metodo alla banda z che ha il rapporto segnale/rumore (S/N) più alto per il candidato, sono stati classificati tutti i target primari e secondari come estesi o puntiformi. Questa classificazione non è stata assegnata ai candidati più deboli, poiché la tecnica non è affidabile a livelli di flusso molto bassi ($z \gtrsim 25.2$mag).

Per ottenere misurazioni del redshift fotometrico, è stata costruita la SED dei candidati LBG, utilizzando una fotometria multi-banda con apertura fissa corretta per il seeing diverso nelle varie bande, estendendo il più possibile l'intervallo di lunghezze d'onda considerato. Pertanto,

oltre alle nuove osservazioni in banda Y e J ottenute con WIRCam, sono state utilizzate le misurazioni fotometriche ottiche in banda r, i e z. Misurazioni fotometriche nelle bande H e K sono state cercate nella survey MUSYC, che copre una vasta area comprendente l'intero campo osservato, fino a un limite tipico di 5σ di $K_{AB} = 21.7$ (Blanc et al. 2008).

Infine, è stato consultato l'archivio Spitzer Heritage Archive (SHA) per immagini IRAC nelle bande 3.6 e 4.5μm per estendere la SED fotometrica a lunghezze d'onda maggiori.

I redshift fotometrici si basano su un procedimento di fitting basato sul χ^2, applicato ai flussi osservati (o magnitudini) inclusi gli errori fotometrici. Utilizzando la classificazione morfologica come criterio di esclusione, sono stati eliminati i template stellari nella stima del redshift fotometrico nel caso di sorgenti estese. Infatti, mentre le brown dwarfs appaiono sempre non risolte, le galassie Lyman break possono apparire spazialmente estese nelle immagini da terra. La SED viene confrontata con template di galassie, convoluti con la risposta dei filtri di ciascuna banda di input, e corretti per l'assorbimento da parte del mezzo intergalattico, seguendo la prescrizione di Madau (1995). Inoltre, è stato tenuto conto un possibile arrossamento interno assumendo la legge di estinzione di Calzetti (Calzetti et al. 2000). Se un oggetto non è rilevato in una banda, il suo flusso è impostato a F_{lim} e il suo errore a 1σ è pari a $F_{lim}/2$, dove F_{lim} è il flusso corrispondente alla magnitudine limite a 2σ in quella banda. Per gli oggetti rilevati solo nella banda z, è stata utilizzata la misura del redshift fotometrico come controllo incrociato con le informazioni ottenute dal diagramma colore-colore $z - Y$ vs $i - z$.

In questo lavoro di tesi, il catalogo è stato impiegato come test set per validare alcune fasi del lavoro. Anche in questo caso, sono state selezionate le magnitudini nelle bande i, Y e z, ma le classi presenti sono limitate a:

- Brown Dwarf: 16 oggetti,
- Galassie: 16 oggetti.

2.4 Analisi visiva dei dati

Per ottenere una visione complessiva della distribuzione dei dati, sono stati analizzati gli **scatter plot** delle magnitudini per il dataset di **SHELLQs** e di **J1030** (Figura 2.2). Questi grafici consentono di esplorare la relazione tra le diverse magnitudini, evidenziare eventuali differenze tra le classi (Quasar, Galassie, Brown Dwarf) e i due dataset. In aggiunta agli scatter plot, è stata applicata la **Kernel Density Estimation (KDE)** per mostrare la **distribuzione** dei dati per ciascuna magnitudine. La **KDE** è una tecnica non parametrica utilizzata per stimare la funzione di densità di probabilità di una variabile casuale continua. La **KDE** illustra dove si concentrano maggiormente gli oggetti astronomici per le diverse classi e per le diverse magnitudini. Ad esempio, se si osserva un picco nelle aree di una determinata magnitudine, ciò

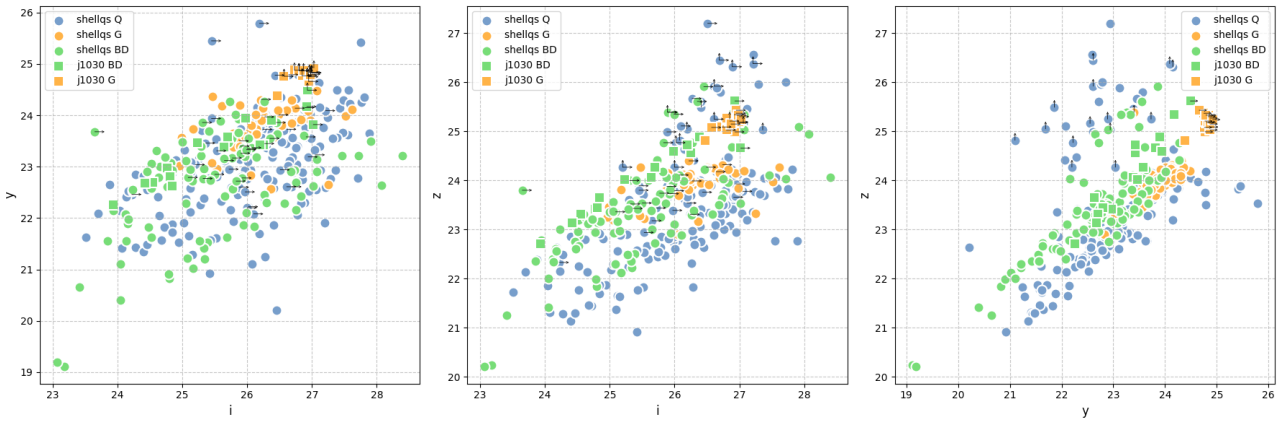
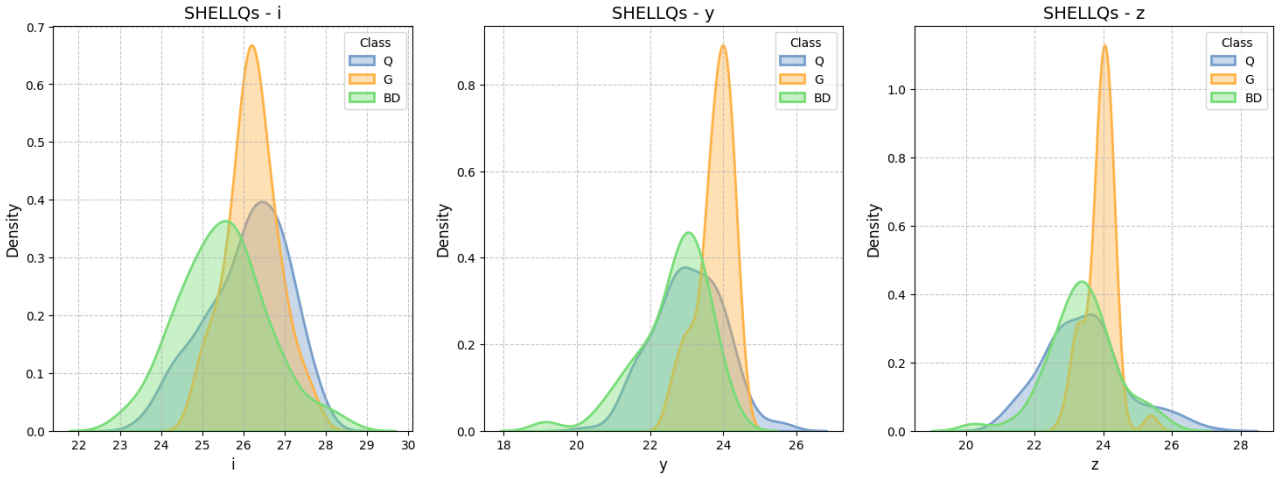
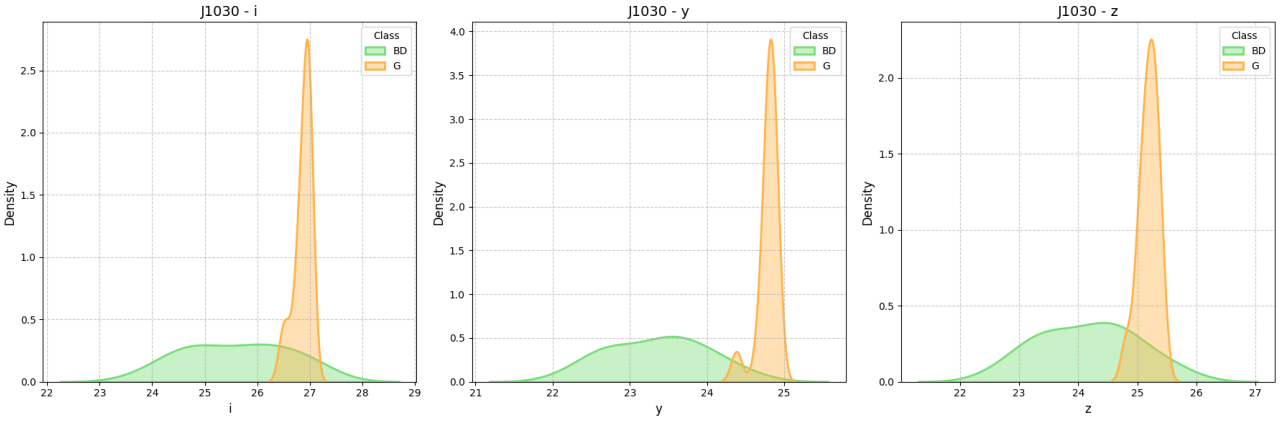


Figura 2.2: Scatter plot delle magnitudini per i dataset SHELLQs e J1030. I cerchi rappresentano gli oggetti appartenenti al catalogo SHELLQs, mentre i quadrati indicano quelli del catalogo J1030. I quasar sono evidenziati in blu, le brown dwarfs in verde e le galassie in arancione. Le frecce indicano i valori di magnitudine riportati come lower limits.



(a)



(b)

Figura 2.3: Kernel Density Estimation (KDE) per il dataset SHELLQs (a) e J1030 (b). In blu la distribuzione dei quasar, in verde quella delle brown dwarf e in arancione le galassie.

può suggerire una maggiore concentrazione di oggetti di una classe specifica in quella regione dello spazio delle caratteristiche.

Dagli scatter plot (Figura 2.2) e dalle distribuzioni KDE (Figura 2.3(a)) osserviamo che i quasar sono ampiamente distribuiti nello spazio delle magnitudini.
In questi grafici, le brown dwarfs nei dataset SHELLQs e J1030 tendono ad occupare le stesse regioni, avendo esse un'ampia distribuzione per entrambi i dataset; in particolare, dal KDE di J1030 (Figura 2.3(b)) si nota l'assenza di un picco significativo, probabilmente dovuto al fatto che J1030 è ad alta latitudine galattica e quindi osserva stelle di alone galattico che sono più distribuite in distanza e conseguentemente in magnitudine.
Le galassie, invece, come evidenziato dalle distribuzioni KDE (Figura 2.3) che mostrano un picco molto pronunciato e una distribuzione più stretta, tendono ad avere distribuzioni meno ampie. Inoltre, possiamo notare dagli scatter plot che le galassie di J1030 tendono ad avere magnitudini più alte rispetto a quelle di SHELLQs occupando di conseguenza zone differenti nello spazio delle magnitudini. Infatti, per queste galassie si osservano numerosi lower limits nelle magnitudini i e Y.
Questa discrepanza è dovuta alla diversa natura delle survey: SHELLQs è progettata per individuare quasar, oggetti tipicamente più luminosi, mentre J1030 è focalizzata sulla ricerca di galassie ad alto redshift più deboli in termini di magnitudine. L'apparente compattezza delle galassie nello spazio delle magnitudini in J1030 potrebbe rappresentare una sfida nella valutazione su dati complessi, in quanto non riflette la maggiore diversità osservata in SHELLQs. Queste differenze evidenziano l'importanza di considerare attentamente la natura dei dati di addestramento e test al fine di ottenere una valutazione accurata delle performance del modello.

2.4.1 Calcolo e analisi visiva dei colori

Un passo fondamentale nel preprocessing dei dati è stato il calcolo dei colori degli oggetti, che fornisce informazioni cruciali sulla loro distribuzione nello spazio delle caratteristiche. I colori sono definiti come la differenza tra le magnitudini osservate nelle varie bande fotometriche e sono utilizzati per caratterizzare meglio le classi di oggetti astronomici. In particolare, sono stati calcolati i seguenti colori:

- $i - z$
- $i - Y$
- $z - Y$

Per ottenere una visione d'insieme della distribuzione dei dati, sono stati analizzati, anche per i colori, gli **scatter plot** e le stime di densità kernel (**KDE**) dei colori ($i - Y$, $z - Y$, $i - z$)

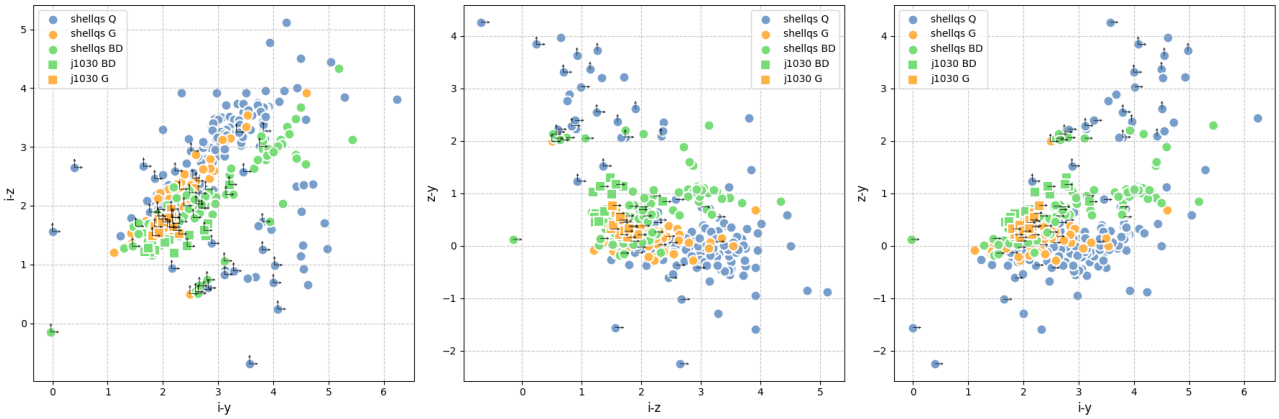
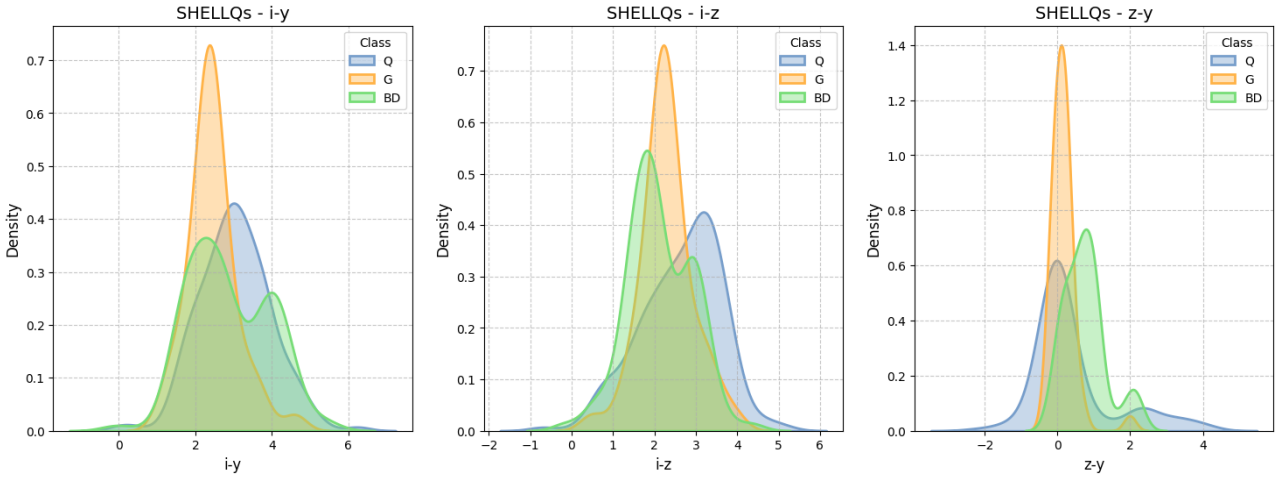
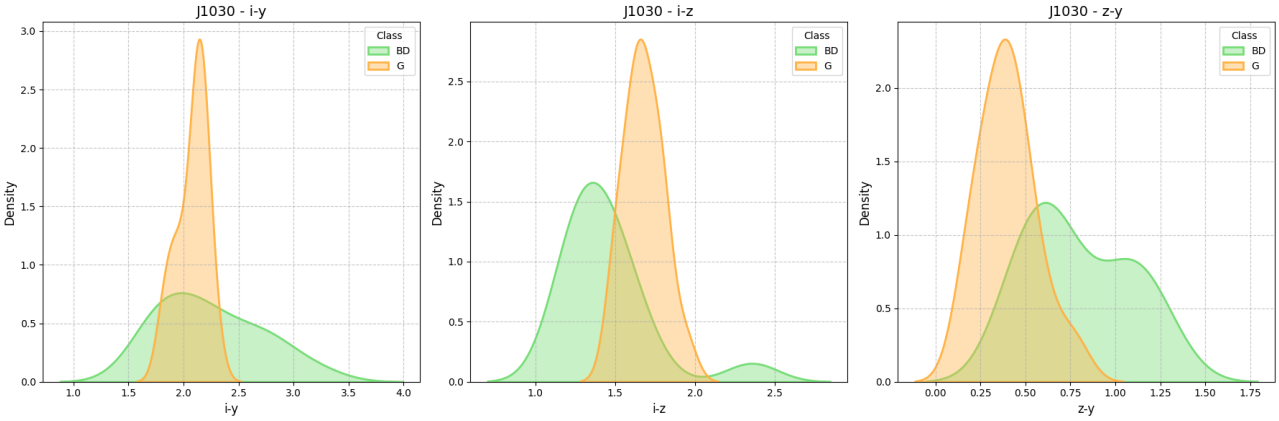


Figura 2.4: Scatter plot dei colori per i dataset SHELLQs e J1030. I cerchi rappresentano gli oggetti appartenenti al catalogo SHELLQs, mentre i quadrati indicano quelli del catalogo J1030. I quasar sono evidenziati in blu, le brown dwarfs in verde e le galassie in arancione. Le frecce indicano i valori dei colori riportati come lower limits.



(a)



(b)

Figura 2.5: Kernel Density Estimation (KDE) per i colori del dataset SHELLQs (a) e J1030 (b). In blu la distribuzione dei quasar, in verde quella delle brown dwarf e in arancione le galassie.

nei due dataset, **SHELLQs** e **J1030**.

Dall'analisi degli scatter plot (Figura 2.4) emerge che i quasar presentano una distribuzione ampia per la maggior parte dei colori, ad eccezione del colore $z - Y$, in cui mostrano una distribuzione più concentrata. Questo comportamento è ulteriormente confermato dal KDE, che evidenzia un picco in corrispondenza di tale valore.
In generale, gli scatter plot mostrano che, sebbene galassie, brown dwarfs e quasar tendano a sovrapporsi, si raggruppano in cluster distinti, rendendoli facilmente distinguibili. Questo comportamento è atteso, poiché le galassie e le brown dwarfs mostrano colori differenti, come già trattato nel Capitolo 1. Inoltre, i quasar presentano un numero significativamente maggiore di punti che si discostano dalla distribuzione principale (Figura 2.4).
Per quanto riguarda le brown dwarfs, i loro colori mostrano una buona sovrapposizione tra i due dataset. Tuttavia, nel dataset SHELLQs la distribuzione risulta più ampia rispetto a quella osservata in J1030.
Infine, le galassie presentano una distribuzione più concentrata in entrambi i dataset, con una discreta sovrapposizione tra SHELLQs e J1030. Tuttavia, è importante notare come nel dataset J1030 siano presenti numerosi upper limits per le coppie di colori $i - z$ e $i - Y$, e lower limits per $z - Y$. Questa differenza nella distribuzione, come già accennato nel capitolo precedente, è in parte dovuta agli obiettivi delle due survey: SHELLQs è progettata per identificare quasar, selezionando oggetti più luminosi, mentre J1030 è focalizzata sulla ricerca di LBG, che sono generalmente più deboli.

Le differenze tra i due dataset devono essere attentamente considerate nella fase di suddivisione dei dati, al fine di garantire una valutazione accurata delle performance del modello. Inizialmente, il modello è stato addestrato utilizzando esclusivamente il dataset SHELLQs. Tuttavia, per ottenere una valutazione più realistica e rappresentativa, il dataset SHELLQs è stato successivamente suddiviso in training e validation set, mentre il dataset J1030 è stato utilizzato come test set. Questa scelta è motivata dal fatto che J1030 presenta caratteristiche più simili ai dati che l'algoritmo dovrà classificare nelle applicazioni future. Nel paragrafo seguente verrà illustrata nel dettaglio la strategia adottata per questa suddivisione.

Capitolo 3

Algoritmi di Machine Learning

Nel contesto dell'apprendimento automatico, la classificazione è un compito fondamentale che consiste nell'assegnare delle classi a oggetti in base alle loro proprietà, o caratteristiche. In questo capitolo, esploreremo il funzionamento dei principali algoritmi utilizzati nel progetto di tesi per classificare oggetti astronomici, come galassie, stelle e quasar, sulla base di dati osservativi. Sebbene nel progetto finale siano stati utilizzati principalmente gli algoritmi **Random Forest** (RF), **Linear Discriminant Analysis** (LDA) e **Probabilistic Random Forest** (PRF), è importante includere una descrizione dei **Decision Trees**, poiché questi ultimi costituiscono la base di funzionamento sia del Random Forest che del Probabilistic Random Forest. I **Decision Trees** sono modelli di classificazione che utilizzano una sequenza di domande per suddividere i dati in base alle caratteristiche, creando così un albero decisionale. Comprendere come funziona un singolo albero decisionale è fondamentale per comprendere come gli Algoritmi di Ensemble, come il Random Forest e il Probabilistic Random Forest, migliorano le performance dei modelli aggregando diversi alberi decisionali. Gli algoritmi trattati in questo capitolo includono:

- **Decision Trees**, che utilizzano una sequenza di domande per suddividere i dati in base a determinate caratteristiche.
- **Random Forest**, un metodo che aggrega più alberi decisionali per migliorare la robustezza e la precisione del modello.
- **Linear Discriminant Analysis** (LDA), un metodo lineare per massimizzare la separazione tra le classi in spazi a bassa dimensione, cioè spazi in cui i dati sono rappresentati utilizzando un numero limitato di caratteristiche.
- **Probabilistic Random Forest**, una versione avanzata del Random Forest che integra le incertezze probabilistiche nei dati e nelle etichette.

Per le definizioni dei termini tecnici utilizzati in questo capitolo, si rimanda al **Vocabolario dei Termini Tecnici** in Appendice A.

Ogni algoritmo sarà descritto evidenziando il suo funzionamento teorico.

3.1 Decision Trees

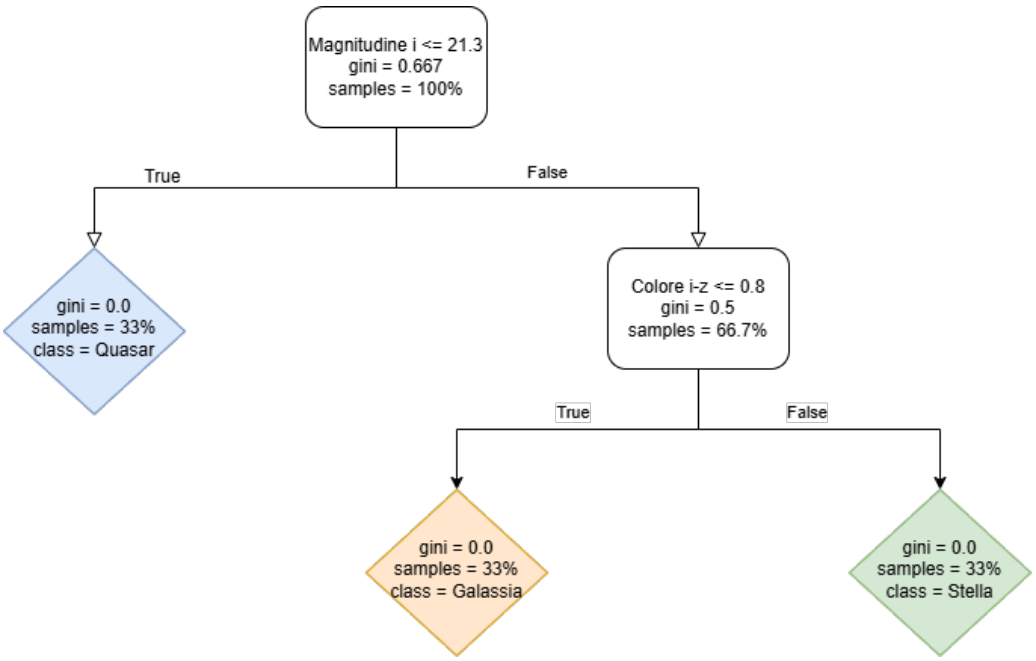


Figura 3.1: Un esempio diagramma che rappresenta un albero decisionale con nodi decisionali e foglie. Ogni nodo presenta una domanda su una caratteristica del dataset (ad esempio, magnitudini o colori fotometrici), suddividendo i dati in base a una soglia specifica. I valori associati a ciascun nodo mostrano l'indice di Gini, che misura l'impurità del nodo: un valore basso indica una maggiore purezza (ovvero, i dati appartengono principalmente a una sola classe). Le foglie finali, rappresentate come box colorate, rappresentano le classi predette (Galassia, Stella, Quasar) nel nostro esempio, riflettendo la decisione finale del modello.

Un **Decision Tree** è un modello di apprendimento supervisionato che prende decisioni basate su una sequenza di domande (**nodi**) relative alle caratteristiche dei dati. Ogni nodo dell'albero rappresenta una domanda (o una condizione) su una caratteristica, mentre ogni **foglia** rappresenta una classe o un valore di output (Figura 3.1). Supponiamo di voler classificare oggetti astronomici in Galassie, Stelle e Quasar utilizzando le loro magnitudini osservate in diverse bande fotometriche (i, Y, z). Un Decision Tree, generato come risultato dell'addestramento del modello, potrebbe iniziare con la domanda:

- La magnitudine i è minore di 21.3?
  Se sì, l'oggetto potrebbe essere un quasar. Altrimenti, il nodo successivo potrebbe chiedere:

- Il colore $i - z$ è minore o uguale a 0.8?
  Questo processo continua fino a raggiungere una foglia che rappresenta una classe specifica, come Galassia.

Seguendo questa procedura, l'albero in Figura 3.1 classifica il 33% degli oggetti come Quasar, Galassie e Stelle.

3.1.1 Principi fondamentali del Decision Tree

Per ogni nodo dell'albero viene scelta una caratteristica e un valore soglia che dividerà i dati in due gruppi. L'obiettivo è scegliere la caratteristica e la soglia che meglio separano i dati, massimizzando la purezza dei sottoinsiemi creati. L'**impurità** di un nodo rappresenta il grado di mescolanza delle classi al suo interno. Un nodo con elevata impurità contiene oggetti appartenenti a diverse classi in proporzioni simili, mentre un nodo con bassa impurità è composto principalmente da oggetti appartenenti a una sola classe (Breiman et al. (1986)). Per esempio, il primo nodo in Figura 3.1 ha impurità (indicata con il termine `gini`) massima in quanto costituito da oggetti appartenenti a 3 classi differenti distribuite in uguale proporzione, il secondo nodo ha impurità inferiore in quanto sono presenti solo due delle tre classi, mentre le foglie hanno impurità zero in quanto in ognuna di esse sono presenti solo oggetti appartenenti ad una classe.
Un buon modello di Decision Tree cerca di ridurre progressivamente l'impurità con ogni suddivisione, aumentando la purezza dei nodi figli. Per misurare l'impurità, vengono utilizzati criteri matematici che valutano la distribuzione delle classi in ciascun nodo. Questi criteri includono:

- **indice di Gini**: misura quanto spesso due oggetti scelti a caso dal nodo appartengono a classi diverse. Valori bassi indicano che gli oggetti appartengono per lo più alla stessa classe, quindi il nodo è puro. Gini tende a dare maggiore enfasi alla classe più dominante, cercando di massimizzare la separazione tra le classi.

- **entropia**: deriva dalla teoria dell'informazione e misura il livello di incertezza nella distribuzione delle classi nel nodo. È più sensibile alla presenza di classi meno rappresentate, poiché considera tutte le classi e il grado di incertezza associato. Valori elevati indicano maggiore disordine o incertezza.

Entrambi i criteri tendono a portare a risultati simili, ma presentano leggere differenze nel modo in cui valutano la purezza. L'indice di Gini enfatizza la separazione delle classi, mentre

l'entropia è più sensibile a distribuzioni con classi meno rappresentate (Breiman et al. (1984); Quinlan (1986)).
Si immagini di avere un nodo con 90 oggetti appartenenti alla classe A e 10 oggetti appartenenti alla classe B.

- L'indice di Gini sarà basso (0.18), indicando una forte predominanza della classe A. Questo suggerisce che il nodo è abbastanza puro e che le istanze sono ben separate.
- L'entropia, invece, avrà un valore più alto (0.46), perché considera l'incertezza introdotta dalla presenza della classe meno rappresentata (B). Se il nodo contenesse solo oggetti di classe A, l'entropia sarebbe zero, mentre la presenza anche di pochi oggetti di classe B aumenta l'incertezza complessiva.

In fase di training, ogni nodo esamina tutte le possibili suddivisioni degli oggetti rispetto alle caratteristiche disponibili, calcolando l'impurità per ogni suddivisione possibile. Il modello non sceglie soglie fisse, ma le calcola dinamicamente provando tutte le possibili soglie tra i valori presenti nei dati e selezionando quella che massimizza la purezza dei nodi. Viene quindi scelta la combinazione di caratteristica e soglia che minimizza l'impurità, cercando di ottenere una divisione che produca i nodi figli con la maggiore purezza possibile. Per esempio, supponiamo di avere un dataset come quello riportato sotto:

Oggetto	z	$z - Y$	Classe
A	21.5	0.4	Quasar (QSO)
B	22.8	0.3	Quasar (QSO)
C	23.5	0.7	Galassia (G)
D	22.2	0.4	Galassia (G)

I valori di z nel dataset sono $21.5, 22.2, 22.8, 23.5$, quindi le soglie candidate si trovano tra questi valori. Il modello testa le seguenti soglie:

- Soglia $z = 22.0$:
  - Nodo sinistro ($z < 22.0$): $\{A$ (QSO)$\}$
    $Gini = 0$
  - Nodo destro ($z \geq 22.0$): $\{B$ (QSO)$, C$ (G)$, D$ (G)$\}$
    $Gini = 0.44$

  Impurità totale: 0.33
- Soglia $z = 22.5$:
  - Nodo sinistro ($z < 22.5$): $\{A$ (QSO)$, D$ (G)$\}$
    $Gini = 0.5$

– Nodo destro ($z \geq 22.5$): $\{B$ (QSO), C (G)$\}$
$Gini = 0.5$

Impurità totale: 0.5

- Soglia $z = 23.0$:

  – Nodo sinistro ($z < 23.0$): $\{A$ (QSO), B (QSO), D (G)$\}$
  $Gini = 0.44$

  – Nodo destro ($z \geq 23.0$): $\{C$ (G)$\}$
  $Gini = 0$

  Impurità totale: 0.33

Tra queste soglie, le migliori sono $z = 22.0$ e $z = 23.0$, che portano a un'impurità totale di 0.33, inferiore rispetto all'altra soglia testata. L'impurità totale rappresenta la riduzione di impurità (Appendice A), rispetto al nodo padre e viene utilizzata per determinare la suddivisione ottimale. I valori di $z - Y$, invece, sono $0.3, 0.4, 0.7$, quindi le soglie candidate sono tra questi valori:

- Soglia $z - Y = 0.35$:

  – Nodo sinistro ($z - Y < 0.35$): $\{B$ (QSO)$\}$
  $Gini = 0$

  – Nodo destro ($z - Y \geq 0.35$): $\{A$ (QSO), C (G), D (G)$\}$
  $Gini = 0.44$

  Impurità totale: 0.33

- soglia $z - Y = 0.5$:

  – Nodo sinistro ($z - Y < 0.5$): $\{A$ (QSO), B (QSO)$\}$
  $Gini = 0$

  – Nodo destro ($z - Y \geq 0.5$): $\{C$ (G), D (G)$\}$
  $Gini = 0$

  Impurità totale: 0

La suddivisione con $z - Y = 0.5$ porta a nodi completamente puri (impurità totale = 0). Poiché il modello seleziona la suddivisione che minimizza maggiormente l'impurità, sceglierà la caratteristica $z - Y$ con soglia 0.5, poiché porta a impurità nulla, rispetto alla migliore suddivisione su z, che ha impurità 0.33.

L'operazione di divisione viene ripetuta finché:

1. Si raggiunge una profondità massima dell'albero, corrispondente al numero massimo di domande che è possibile formulare. Questa può essere definita all'inizio, come parametro del modello, oppure determinata in corso d'opera, lasciando che l'albero cresca fino al soddisfacimento di specifici criteri di arresto.

2. Tutti gli oggetti in un nodo sono della stessa classe (purezza massima).

3. Non ci sono altre caratteristiche utili per effettuare nuove divisioni.

Una volta che l'albero ha effettuato tutte le divisioni, arriviamo a una foglia, che rappresenta la classe finale per gli oggetti che raggiungono quel nodo.
Ad esempio, se una foglia contiene 7 Galassie e 3 Stelle, il modello assegnerà agli oggetti in quella foglia la classe Galassia perché è la classe dominante. L'affidabilità della classificazione dipenderà dall'impurità del nodo foglia.
A questo punto la fase di training finisce e il modello migliore verrà applicato ai campioni da classificare.

3.1.2 Limitazioni del Decision Tree

Sebbene il Decision Tree sia efficace e intuitivo, esso presenta alcune limitazioni, come la sensibilità all'overfitting (si veda appendice A) e alla variabilità degli oggetti di training (Breiman et al. (1984)).

Un Decision Tree addestrato su oggetti che presentano caratteristiche con errori o anomalie potrebbe creare nodi molto specifici, come:

- La magnitudine z è esattamente 20.43?

Questi nodi non generalizzano bene su nuovi oggetti, ossia oggetti che non sono stati utilizzati durante l'addestramento e che provengono da distribuzioni simili ma non identiche. Ad esempio, un oggetto con $z = 20.42$ potrebbe essere erroneamente classificato in una classe sbagliata, poiché il modello considera solo la corrispondenza esatta con il valore 20.42 e ignora la variabilità naturale degli oggetti. Questo porta a una scarsa capacità di generalizzazione e a un rischio maggiore di overfitting, dove il modello si adatta troppo ai dettagli delle caratteristiche degli oggetti di addestramento, senza essere in grado di fare previsioni affidabili su nuovi oggetti.

3.2 Random Forest

Il **Random Forest** supera queste limitazioni combinando vari di Decision Tree, in modo che ogni albero contribuisca al risultato finale. In questo modo, il Random Forest bilancia la capacità del Decision Tree di individuare pattern complessi con una maggiore robustezza e capacità di

generalizzazione.

Ogni albero è costruito su un sottoinsieme casuale del dataset, e alla fine gli alberi "votano" per determinare la classe finale. Questo processo è conosciuto come **Bagging** (Bootstrap Aggregation), una tecnica di ensemble che unisce i risultati di modelli deboli, come gli alberi decisionali, per ottenere un modello più robusto.

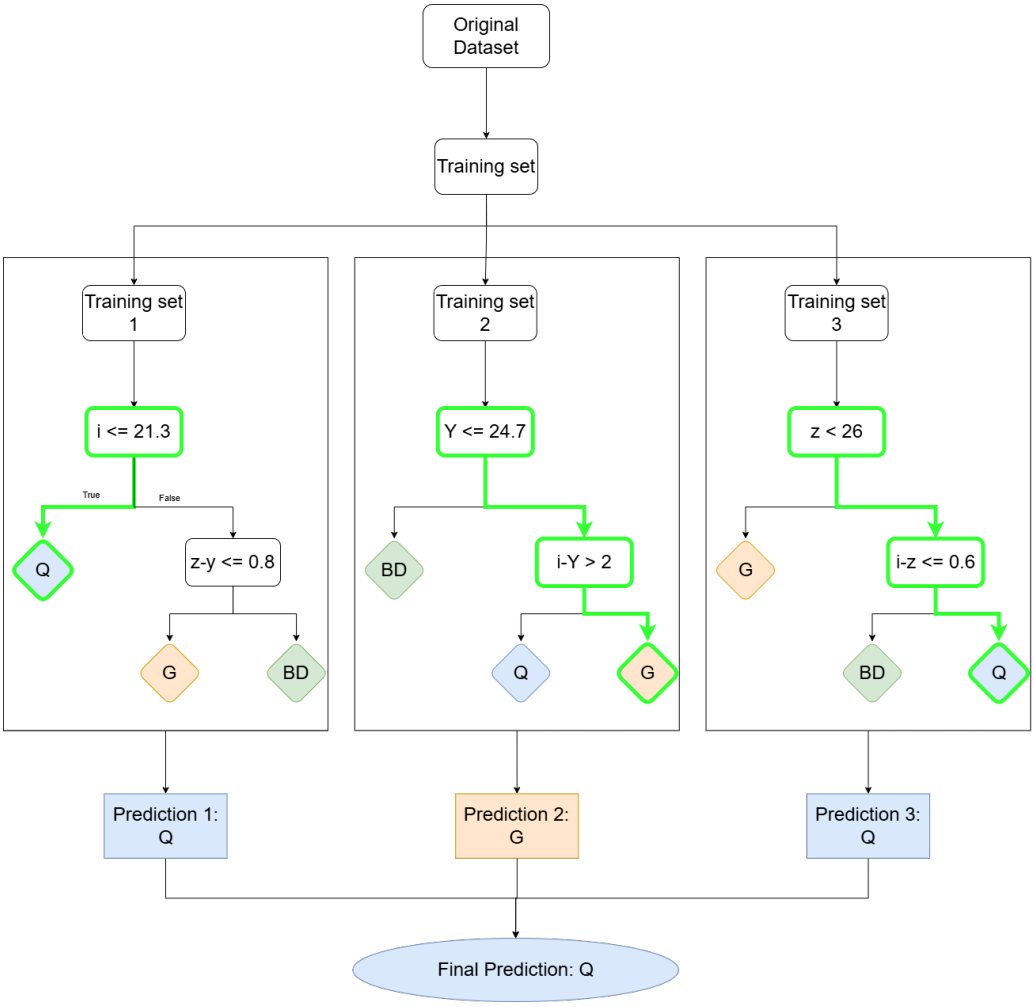


Figura 3.2: Un diagramma che rappresenta il funzionamento di un Random Forest. L'immagine mostra più alberi decisionali costruiti a partire da sottoinsiemi casuali del dataset e delle caratteristiche, evidenziando il processo di votazione a maggioranza per determinare la classe finale. Q = Quasar, G = Galassie, BD = Brown Dwarfs

3.2.1 Funzionamento di base del Random Forest

Per comprendere come il Random Forest operi, è utile seguire i principali step descritti nei lavori di Breiman et al. (2001); Hastie et al. (2009); Cutler et al. (2007). Il processo di creazione del

modello si articola nei seguenti passaggi:

1. **Selezione di un sottoinsieme di oggetti e campionamento bootstrap**: Si seleziona un campione casuale dal dataset, nel caso del campionamento bootstrap, ogni modello viene creato da un campione con reinserimento, consentendo che alcuni oggetti possano essere selezionati più volte, mentre altri potrebbero essere esclusi. La dimensione del campione selezionato è generalmente pari a quella del dataset originale, ma non esiste una frazione minima predefinita del campione iniziale.
   Supponiamo di avere un dataset con 10 oggetti astronomici, etichettati come Galassie o Quasar, identificati dai colori $(i - z)$ e $(z - Y)$. Durante il campionamento *bootstrap* per costruire un albero, possiamo selezionare casualmente gli oggetti: ad esempio, l'insieme $\{3, 5, 1, 3, 7, 5, 8, 2, 9, 1\}$. Alcuni oggetti (come 3 e 5) sono inclusi più volte, mentre altri come (4 e 6) non sono selezionati e rimangono *out-of-bag*.
   Questo assicura varietà tra gli alberi e riduce il rischio di *overfitting*.

2. **Addestramento individuale degli alberi**: Ogni albero viene addestrato sul campione bootstrap generato al passaggio precedente, generando una previsione basata sul suo campione di dati.

3. **Aggregazione dei risultati**: I risultati dei singoli alberi sono combinati per determinare il risultato finale. L'output finale è determinato dalla votazione a maggioranza tra gli alberi per la classificazione.
   Supponiamo che stiamo cercando di classificare un oggetto sconosciuto. I singoli alberi predicono le seguenti classi:

   - Albero 1: Quasar
   - Albero 2: Galassia
   - Albero 3: Quasar

   La votazione a maggioranza classifica l'oggetto come Quasar, dato che questa è la classe predetta da più della metà degli alberi (Figura 3.2).

3.2.2 Randomizzazione delle caratteristiche

Un altro elemento cruciale del Random Forest è la randomizzazione delle caratteristiche: ogni albero è costruito utilizzando un sottoinsieme casuale di oggetti e caratteristiche (Breiman et al. (2001); Hastie et al. (2009)).

Durante la costruzione dell'albero k-esimo, viene generato un vettore casuale k che determina quali caratteristiche verranno prese in considerazione a ciascun nodo. Questo vettore è costituito da una selezione casuale di indici tra le caratteristiche disponibili nel dataset, e la sua lunghezza

è un parametro arbitrario del modello. Generalmente, si sceglie pari alla radice quadrata del numero totale di caratteristiche nel caso di problemi di classificazione. Ogni albero ha un proprio vettore k, che viene generato in modo indipendente rispetto a quelli degli altri alberi.

Sebbene il contenuto di k cambi da un albero all'altro, il processo con cui viene generato segue sempre la stessa distribuzione di probabilità. In altre parole, ogni caratteristica ha la stessa probabilità di essere selezionata in ciascun albero, garantendo così uniformità nel criterio di selezione pur mantenendo la diversità tra gli alberi.

L'albero risultante è quindi un classificatore della forma $h(x, k)$, dove x rappresenta l'input (un oggetto da classificare) e k il vettore delle caratteristiche selezionate casualmente per quell'albero specifico. Questo classificatore viene poi utilizzato per fare previsioni su nuovi esempi.

A ogni nodo, il modello seleziona un sottoinsieme casuale delle caratteristiche disponibili. Questo processo riduce la correlazione tra gli alberi e aiuta a prevenire l'overfitting. Si immagini di avere le caratteristiche $(i - z)$, $(z - Y)$, e Y. Quando il modello costruisce un nodo, seleziona casualmente un sottoinsieme di queste caratteristiche. Ad esempio, potrebbe scegliere solo $(i - z)$ e Y e successivamente valuta quale di queste due caratteristiche porta alla divisione migliore dei dati. Questo processo di selezione casuale viene ripetuto per ogni nodo in ciascun albero. Poiché ogni albero è costruito utilizzando un sottoinsieme diverso di dati, grazie al bootstrap, e ogni nodo seleziona casualmente un sottoinsieme di caratteristiche, gli alberi risultano molto diversificati, anche se addestrati sullo stesso dataset. Questo è ciò che distingue il Random Forest dai Decision Tree tradizionali, che tendono ad adattarsi troppo ai dati di training, con un rischio maggiore di overfitting.

3.2.3 Valutazione Out-of-Bag (OOB)

Durante il processo di bagging, ogni albero viene addestrato su un campione ottenuto con bootstrap, ovvero se il dataset di training contiene N oggetti, allora il campione bootstrap avrà anch'esso dimensione N, ma alcuni oggetti verranno selezionati più volte, mentre altri potrebbero non essere scelti affatto. Ogni oggetto ha una probabilità di non essere scelto in una singola estrazione pari a $1 - \frac{1}{N}$. Dopo N estrazioni, la probabilità che un oggetto non venga mai selezionato è approssimativamente $e^1 \approx 0.37$. Di conseguenza, circa il 37% degli oggetti non viene incluso nel campione di training per un dato albero, mentre il restante 63% viene selezionato almeno una volta. Questa proprietà è indipendente dalla dimensione del dataset e deriva direttamente dal campionamento bootstrap. (Breiman et al. (2001); Hastie et al. (2009); Cutler et al. (2007)).

Gli oggetti esclusi da un albero specifico sono definiti come out-of-bag (OOB). Anche se non vengono utilizzati per l'addestramento di quell'albero, potrebbero comunque essere inclusi nei campioni di training di altri alberi all'interno della Random Forest.

Gli oggetti OOB sono fondamentali per valutare le prestazioni del modello senza la necessità di un set di validazione separato. Poiché questi dati non sono stati utilizzati per addestrare un determinato albero, possono essere impiegati per testarne le prestazioni. La valutazione OOB fornisce una stima dell'errore di generalizzazione, calcolando l'accuratezza del modello sull'intero dataset di training, risparmiando così il set di validazione per altre analisi.

Si immagini un dataset con 100 stelle. Durante il campionamento bootstrap per costruire un albero, circa 37 stelle non verranno selezionate. Questi oggetti OOB possono essere utilizzati per testare quell'albero, calcolando un'accuratezza provvisoria. Ripetendo il processo per tutti gli alberi e aggregando le predizioni sui dati OOB (ossia combinando i risultati dei vari alberi in un'unica predizione tramite maggioranza), otteniamo una stima globale delle prestazioni del modello, senza dover ricorrere a un set di validazione separato. Questo approccio è particolarmente utile quando il numero di dati a disposizione è limitato.

3.2.4 Importanza delle caratteristiche nel Random Forest

Un altro aspetto fondamentale del Random Forest è la capacità di valutare l'importanza delle caratteristiche, permettendo una maggiore interpretabilità del modello. Durante la costruzione di ciascun albero, il modello misura quanto ciascuna caratteristica contribuisca a ridurre l'impurità nei nodi. Questa riduzione di impurità, aggregata su tutti gli alberi della foresta, fornisce una prima stima dell'importanza di ciascuna caratteristica (Breiman et al. (2001); Hastie et al. (2009); Cutler et al. (2007)). La riduzione di impurità calcolata in ogni nodo viene sommata per ciascuna caratteristica, considerando tutti i nodi in cui quella caratteristica è stata utilizzata come criterio di suddivisione in tutti gli alberi della foresta. Il valore totale della riduzione di impurità di ogni caratteristica viene diviso per la somma totale delle riduzioni di impurità di tutte le caratteristiche nella foresta.

Inoltre, l'importanza delle caratteristiche può essere valutata anche utilizzando i dati OOB. Per farlo, si "disturbano" i valori di una caratteristica permutandoli casualmente all'interno del dataset OOB, mantenendo invariati i valori delle altre caratteristiche. Questo processo rompe la relazione originale tra la caratteristica permutata e il target. Successivamente:

- Si calcola l'accuratezza del modello sui dati OOB modificati.
- Si confronta l'accuratezza ottenuta con quella originale (senza permutazione).

Il fatto che l'accuratezza peggiori in modo significativo implica che la caratteristica permutata ha un ruolo cruciale nelle predizioni del modello. Se l'accuratezza rimane invariata, la caratteristica ha un'importanza marginale. Questo approccio, noto come *permutation feature importance*, permette di identificare le caratteristiche più influenti nel modello, facilitando l'interpretazione dei risultati e supportando analisi ulteriori per migliorare la qualità delle predizioni.

Supponiamo di costruire un Random Forest per distinguere tra Galassie e Brown Dwarf utilizzando le caratteristiche $(i - z)$, $(z - Y)$ e Y. Durante il training, il modello scopre che la differenza $(z - Y)$ contribuisce maggiormente a ridurre l'impurità nei nodi rispetto alle altre caratteristiche. Questo indica che $(z - Y)$ è una caratteristica importante per distinguere le classi. Inoltre, disturbando i valori di $(z - Y)$ nei dati OOB, l'accuratezza del modello diminuisce drasticamente, confermando l'importanza della caratteristica.

3.2.5 Limiti e vantaggi

A differenza dei singoli alberi decisionali (*Decision Tree*), che tendono a suddividere ripetutamente lo spazio delle caratteristiche in regioni sempre più piccole e specifiche per adattarsi perfettamente ai dati di training, il Random Forest riduce questa tendenza aggregando alberi diversi. Questo consente di ottenere un modello più bilanciato, che riduce la varianza senza aumentare il bias. Infatti, aumentando il numero di alberi, l'errore di generalizzazione non peggiora, anzi, dopo un certo numero di alberi, l'errore si stabilizza su un valore limite (Breiman et al. (2001)).

L'aggiunta di alberi rende il Random Forest meno suscettibile all'overfitting rispetto a un singolo Decision Tree. Inoltre, grazie alla valutazione OOB e alla possibilità di analizzare l'importanza delle caratteristiche, il Random Forest permette una migliore comprensione delle variabili che influenzano le predizioni, consentendo una maggiore interpretabilità e un controllo sull'accuratezza generale del modello. In un singolo Decision Tree, un oggetto del set di training con valori estremi delle caratteristiche potrebbe causare una divisione eccessivamente specifica nel nodo corrispondente. Questo porta il modello ad adattarsi troppo strettamente ai dati di training, limitandone la capacità di generalizzare a nuovi dati. Al contrario, in un Random Forest, altri alberi costruiti utilizzando campioni diversi (grazie al campionamento *bootstrap*) e selezioni casuali delle caratteristiche bilanciano l'effetto di eventuali suddivisioni eccessivamente specifiche. Questo processo consente di ottenere un modello complessivo più robusto, in grado di generalizzare meglio su dati non visti.

Questa combinazione di robustezza, capacità di generalizzazione e valutazione diretta dell'importanza delle variabili rende il Random Forest uno strumento efficace per problemi complessi e con dati rumorosi.
Tuttavia, quando il problema richiede una maggiore comprensione della separabilità delle classi in spazi a bassa dimensione, o quando è cruciale ridurre la dimensionalità mantenendo la capacità di distinguere le classi, metodi lineari come l'**Analisi Discriminante Lineare** (LDA) possono fornire un'analisi complementare, offrendo una visione diversa dei dati, in quanto l'LDA cerca una proiezione lineare che massimizza la separazione tra le classi, semplificando la struttura dei dati.

3.3 Linear Discriminant Analysis

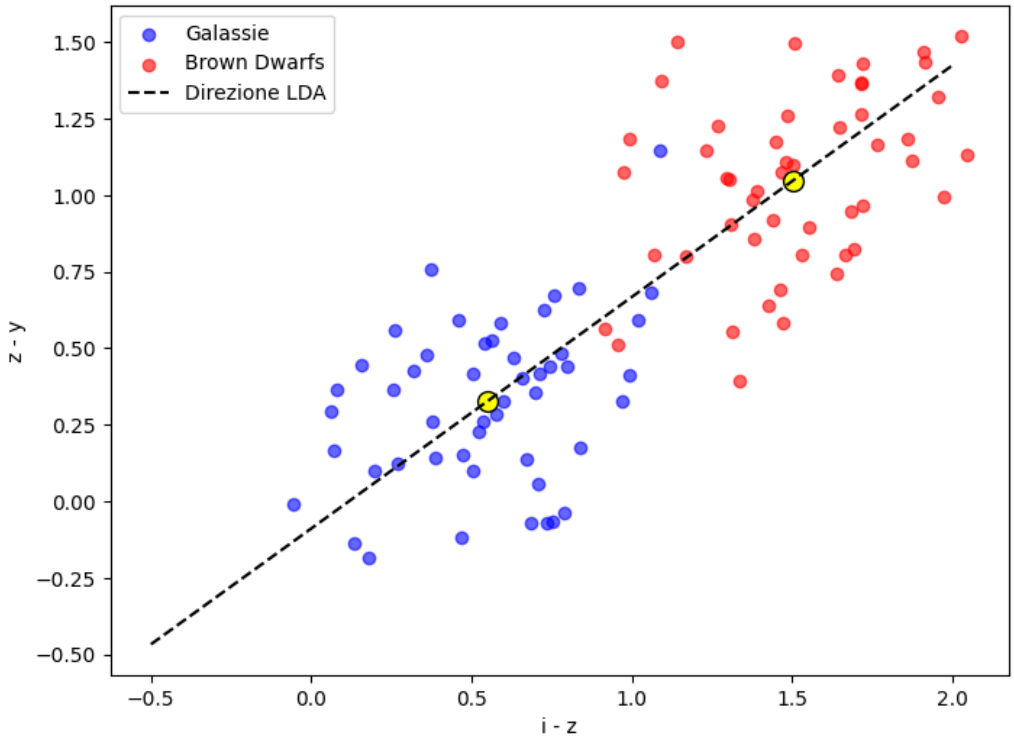


Figura 3.3: Visualizzazione della Linear Discriminant Analysis (LDA) in uno spazio bidimensionale con due classi. La linea tratteggiata rappresenta l'asse discriminante calcolato dall'LDA, che massimizza la separazione tra le classi e riduce la varianza all'interno di ciascuna classe. I punti gialli rappresentano i centroidi delle due classi.

L' **LDA** è una tecnica statistica ampiamente utilizzata sia per la classificazione che, in modo ancora più frequente, per la riduzione della dimensionalità (Fisher et al. (1936); Duda et al. (2001)). Il suo principio fondamentale consiste nel trovare una proiezione lineare dei dati che massimizzi la separazione tra le classi, riducendo al contempo la varianza all'interno di ciascuna classe. Questa proprietà la rende particolarmente utile quando le classi da distinguere sono ben separate nello spazio delle caratteristiche. Possiamo osservare questo principio in Figura 3.3, dove consideriamo due classi di oggetti astronomici nello spazio bidimensionale $(i - z,\ z - Y)$. Gli oggetti della classe A (es. galassie) si concentrano attorno a (0.5, 0.3), mentre quelli della classe B (es. brown dwarf) intorno a (1.5, 1.0); punti gialli in Figura 3.3. L'LDA cerca una funzione lineare di entrambe le variabili che massimizzi la distanza tra le proiezioni di questi oggetti delle due classi (Figura 3.3).

3.3.1 Assunzioni teoriche dell'LDA

LDA si basa su alcune assunzioni teoriche fondamentali, che influenzano il suo funzionamento (Bishop et al (2006)):

- **Normalità multi variata**: Le caratteristiche indipendenti devono seguire una distribuzione normale per ciascuna classe.
- **Omogeneità di varianza/covarianza (omoschedasticità)**: si richiede che le varianze e le covarianze siano uguali tra i gruppi. Questa condizione può essere verificata utilizzando il **test di Box M** (per maggiori dettagli sul Test di Box M, si veda l'Appendice B), un test statistico che confronta le matrici di covarianza per determinare se sono omogenee tra i gruppi. In alternativa, si suggerisce di preferire l'analisi discriminante quadratica (QDA) quando le covarianze non sono uguali.

- **Indipendenza**: è necessario che i valori di una caratteristica per un oggetto siano indipendenti dai valori della stessa caratteristica per altri oggetti nel dataset. Inoltre, gli oggetti, con le loro caratteristiche, devono essere selezionati casualmente.
- **Dimensione della classe più piccola**: la classe con il numero minimo di oggetti ne deve contenere in numero maggiore rispetto a quello di caratteristiche predittive, per garantire risultati affidabili. Supponiamo di avere 3 caratteristiche predittive (es. $i-Y$, $i-z$, $z-Y$) e che la classe con il minor numero di oggetti nel dataset ne abbia 2 (es. 2 brown dwarfs), in questo caso, il numero di oggetti (2) è inferiore al numero di caratteristiche (3). Questo porta a problemi di affidabilità perché il modello non ha abbastanza dati per stimare in modo robusto le relazioni tra le caratteristiche nella classe.
- **Sensibilità agli outlier**: l'LDA è particolarmente sensibile agli outlier, che possono influenzare le stime delle medie e delle covarianze, riducendo l'efficacia del modello.

Nonostante queste assunzioni, l'analisi discriminante lineare si è dimostrata relativamente robusta a lievi violazioni, risultando comunque efficace anche in presenza di variabili dicotomiche, dove la normalità multi variata non sempre è rispettata (Murphy et al. (2012)).

Per affrontare il problema della classificazione, l'**LDA** assume che le funzioni di densità di probabilità condizionali delle caratteristiche x date le classi $y = 0$ e $y = 1$, rispettivamente $p(x|y = 0)$ e $p(x|y = 1)$, siano entrambe distribuzioni normali con parametri di media e covarianza (μ_0, Σ_0) e (μ_1, Σ_1), rispettivamente.
La soluzione ottimale, secondo la regola di Bayes, prevede l'assegnazione di un campione alla seconda classe se il logaritmo del rapporto di verosimiglianza supera una certa soglia T, tale che:

$$\frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0) + \frac{1}{2}ln|\Sigma_0| - \frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) + \frac{1}{2}ln|\Sigma_1| > T$$

Senza ulteriori assunzioni, il classificatore è noto come QDA.

L'LDA, invece, introduce l'ulteriore semplificazione dell'omocedasticità, ovvero si assume che le covarianze delle classi siano identiche ($\Sigma_0 = \Sigma_1 = \Sigma$) e che le covarianze abbiano rango completo. In questo caso, diversi termini si cancellano:

$$x^T \Sigma_0^{-1} = x^T \Sigma_1^{-1} x$$

e il criterio di decisione diventa una soglia sul prodotto scalare:

$$w^T x > c$$

per una costante soglia c, dove:

$$w = \Sigma^{-1}(\mu_1 - \mu_0)$$
$$c = \frac{1}{2} w^T (\mu_1 + \mu_0)$$

Questo significa che il criterio per assegnare una classe y a un input x dipende esclusivamente da una combinazione lineare delle osservazioni note.

Immaginiamo di voler distinguere tra stelle e galassie in un catalogo astronomico basandoci su parametri come magnitudini i, Y e z. L'LDA calcola una combinazione lineare di queste magnitudini, $LDA_{score} = 0.6 \times i + 0.3 \times Y + 0.1 \times z - 0.5$, e stabilisce una soglia, $soglia = -\frac{intercetta}{||w||} \approx -\frac{0.5}{0.68} \approx 0.74$, a questo punto se la media dei LDA_{score} per le galassie è più alta rispetto a quella delle stelle, allora assegniamo la classe "galassia" ai valori superiori alla soglia e la classe "stella" a quelli inferiori. Se invece la media delle stelle fosse più alta, allora faremmo il contrario. Quindi, se $\overline{LDA}_{score}^{galassie} = 1.2$ e $\overline{LDA}_{score}^{stelle} = 0.5$ gli oggetti con $LDA_{score} > 0.74$ saranno classificati come galassie, mentre quelli con $LDA_{score} < 0.74$ saranno classificati come stelle.

Dal punto di vista geometrico, questa conclusione è spesso utile: il criterio per assegnare una classe y ad un input x si riduce a una funzione della proiezione del punto nello spazio multidimensionale x sul vettore w.
In altre parole, l'osservazione appartiene alla classe y se il corrispondente x si trova su un determinato lato di un iperpiano perpendicolare a w. La posizione dell'iperpiano è definita dalla soglia c (Figura 3.4).

3.3.2 Proiezione lineare e separazione delle Classi

L'analisi discriminante opera combinando linearmente le caratteristiche che descrivono gli oggetti. Queste combinazioni lineari generano nuove variabili, dette funzioni discriminanti, che massimizzano la separazione tra le classi. Il numero massimo di funzioni discriminanti che è possibile creare dipende sia dal numero di classi (N_g) sia dal numero di caratteristiche (p): in particolare, si ottengono al massimo $min(N_g - 1, p)$ funzioni discriminanti. Questo significa che,

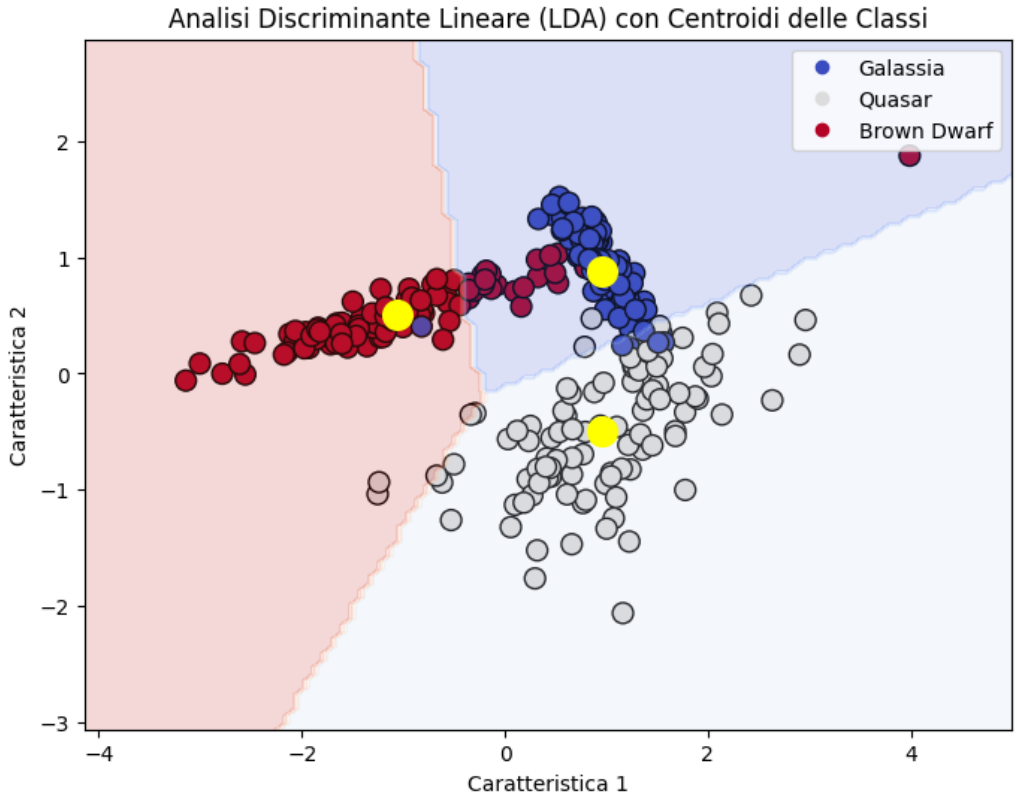


Figura 3.4: Visualizzazione dei dati proiettati tramite l'Analisi Discriminante Lineare (LDA) in uno spazio bidimensionale, con tre classi distinte (Galassie, Quasar, Brown Dwarf). I dati originali sono mostrati come punti colorati, con il colore che rappresenta la classe di appartenenza. La linea di separazione, ottenuta tramite LDA, mostra il confine decisionale che massimizza la separazione tra le classi, riducendo al contempo la varianza all'interno di ciascuna classe. I centroidi delle classi sono indicati con un cerchio giallo, evidenziando la posizione media di ogni classe nello spazio delle caratteristiche. *Caratteristica 1* e *Caratteristica 2* sono combinazioni lineari delle caratteristiche iniziali che sono i, Y e z.

se il numero di caratteristiche è maggiore rispetto a quello delle classi, il numero di funzioni sarà limitato dal numero di classi meno uno; viceversa, se il numero di classi è maggiore, sarà il numero di caratteristiche a rappresentare il limite (Jolliffe et al. (2002); Duda et al. (2001)).

Ogni funzione discriminante ha un ruolo specifico:

- La prima funzione discriminante viene calcolata in modo da massimizzare le differenze tra le classi.
- Le funzioni successive massimizzano ulteriori differenze tra le classi, ma devono essere matematicamente indipendenti (non correlate) dalle funzioni precedenti. Questo processo continua fino a ottenere il numero massimo di funzioni discriminanti.

Per distinguere le classi, l'LDA definisce delle **regioni di classificazione** nello spazio delle funzioni discriminanti. Supponiamo, ad esempio, di avere tre classi di oggetti: galassie, quasar e brown dwarf. L'LDA potrebbe generare una funzione discriminante che separa le galassie dai quasar e una seconda funzione che separa le brown dwarf dalle altre due classi. Ogni classe sarà rappresentata da un **centroide**, che corrisponde al valore medio delle funzioni discriminanti per quella classe. La distanza tra i centroidi indica quanto bene le classi sono separate. L'obiettivo finale è minimizzare l'errore di classificazione, ovvero assegnare correttamente ogni oggetto alla sua classe. (Figura 3.4).

Nel caso in cui ci siano solo due classi viene generata una singola funzione discriminante. Questa funzione rappresenta una linea nello spazio delle caratteristiche che separa le due classi.

Per valutare le funzioni discriminanti, si utilizzano diversi strumenti statistici che permettono di comprendere quanto bene le funzioni separano le diverse classi, fornendo informazioni cruciali sul comportamento del modello. Tra i principali strumenti di valutazione troviamo:

- **Punteggi discriminanti**: sono dei valori calcolati applicando la funzione discriminante agli oggetti. Questi punteggi riflettono la posizione di un oggetto lungo la funzione discriminante e indicano quanto probabilmente un oggetto appartiene a una determinata classe rispetto alle altre.

- **Coefficienti di correlazione strutturale**: misurano quanto ciascuna caratteristica è correlata con il punteggio discriminante di una funzione, considerando ciascuna caratteristica singolarmente. Questi coefficienti aiutano a identificare quali caratteristiche sono più utili per separare le classi.

- **Coefficienti standardizzati**: rappresentano il contributo diretto e unico di ogni caratteristica alla funzione discriminante, tenendo conto delle relazioni tra le caratteristiche (come la correlazione) e normalizzando le scale. Questo è particolarmente utile per identificare quali caratteristiche, tra tutte disponibili, giocano il ruolo più significativo nella funzione discriminante.

- **Centroidi di gruppo**: sono i punteggi medi delle funzioni discriminanti per ciascun gruppo e consentono di valutare la separazione tra le classi: maggiore è la distanza tra i centroidi, minore sarà l'errore di classificazione.

Questi strumenti consentono di identificare le caratteristiche più discriminanti e di valutare l'efficacia complessiva del modello. In particolare, l'analisi della distanza tra i centroidi di gruppo permette di comprendere quanto le classi siano separabili, mentre i punteggi discriminanti e i coefficienti associati offrono una valutazione più dettagliata del contributo di ogni variabile.

L'obiettivo finale dell'LDA consiste nel determinare le regioni ottimali dello spazio campionario per ciascuna classe, al fine di minimizzare l'errore di classificazione. Grazie alla sua capacità di separare le classi in modo chiaro e interpretabile, l'LDA si dimostra particolarmente efficace in contesti di classificazione supervisionata (Jolliffe et al. (2002)). Ad esempio, l'LDA può essere utilizzata per distinguere tra oggetti stellari e non stellari in un catalogo astronomico come la Sloan Digital Sky Survey (SDSS) [1] analizzando parametri come le magnitudini e i colori. Questa tecnica, riducendo la dimensionalità dei dati, permette di identificare con maggiore precisione oggetti peculiari che potrebbero altrimenti risultare difficili da classificare.

3.3.3 Limiti dell'LDA

Sebbene l'LDA e il Random Forest siano tecniche potenti per la classificazione, entrambe non considerano le incertezze associate alle misurazioni e alle etichette. Le incertezze nelle osservazioni astronomiche, come gli errori nelle misure delle magnitudini o le ambiguità nelle etichette di classificazione, sono comuni e possono influire significativamente sulle prestazioni di un modello (Bishop et al. (2006), Murphy et al. (2012)). Per affrontare questa problematica, il **Probabilistic Random Forest** (PRF) rappresenta un approccio avanzato, poiché integra esplicitamente le incertezze nei dati. Questo lo rende particolarmente utile per applicazioni su dati reali, dove le incertezze sono inevitabili e la gestione delle stesse è cruciale per ottenere previsioni affidabili.

3.4 Probabilistic Random Forest

Il PRF è presentato da Reis et al. (2018) sul quale si baserà la seguente trattazione.

Il PRF rappresenta un'estensione del tradizionale algoritmo Random Forest (RF), progettata per migliorare le capacità predittive in presenza di incertezze nei dati. Mentre il Random Forest considera i valori delle caratteristiche e delle classi come deterministici, il PRF tratta questi valori come distribuzioni di probabilità. Questo approccio consente di incorporare le incertezze delle caratteristiche (*feature uncertainty*) e delle classi (*label uncertainty*), migliorando la capacità del modello di affrontare dataset rumorosi e complessi (Figura 3.5). In un'analisi astrofisica, le incertezze nelle magnitudini fotometriche e nella classificazione degli oggetti (Galassie, Stelle, Quasar) sono essere rappresentate come distribuzioni probabilistiche, permettendo al PRF di sfruttare appieno le informazioni disponibili.

[1] La Sloan Digital Sky Survey (SDSS) è un progetto di mappatura del cielo che ha creato una vasta, raccolta, con una copertura che copre 1455 $\deg^2$ di dati astronomici, inclusi cataloghi di galassie, stelle, quasar e altre caratteristiche celesti. Per maggiori dettagli, visitare `https://www.sdss.org/`.

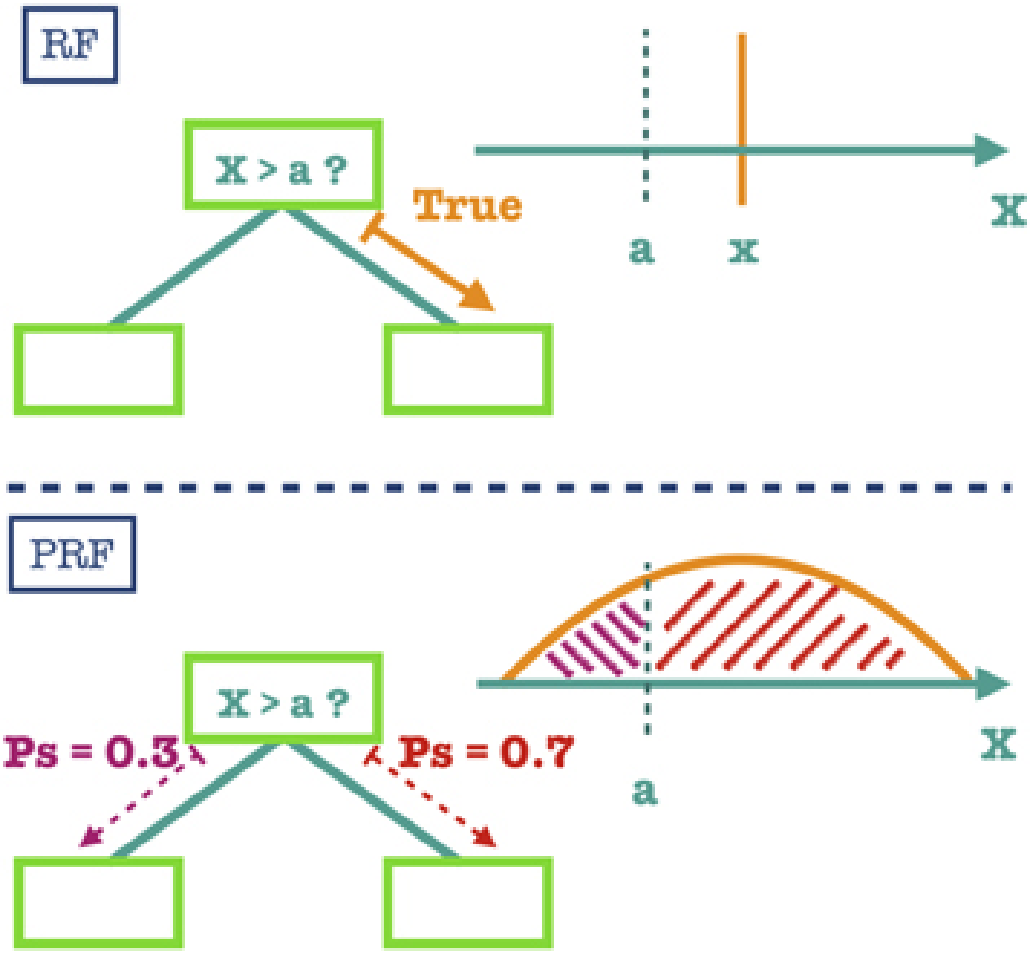


Figura 3.5: Illustrazione della procedura di propagazione degli oggetti nei nodi dell'albero, confrontando il PRF con il RF tradizionale. In entrambi gli algoritmi, ogni nodo dell'albero applica una condizione su una specifica caratteristica dell'oggetto. Nel Random Forest tradizionale, un oggetto viene propagato al ramo destro o sinistro in base al risultato della condizione del nodo (vero = ramo destro, falso = ramo sinistro). Nel Probabilistic Random Forest (PRF), invece, le probabilità di propagazione ($\pi_i(r)$ e $\pi_i(l)$) vengono calcolate tenendo conto dell'incertezza associata al valore della caratteristica dell'oggetto. Di conseguenza, nel PRF l'oggetto viene propagato simultaneamente in entrambi i rami, con probabilità determinate dalla sua incertezza.

3.4.1 Propagazione delle incertezze nel PRF

Il PRF si distingue per la sua capacità di propagare le incertezze attraverso l'intero albero decisionale. A differenza del RF, dove un oggetto segue un singolo percorso nell'albero, nel PRF l'oggetto può propagarsi in più rami contemporaneamente, con probabilità determinate dalle incertezze delle feature (Figura 3.5). Inoltre, il PRF gestisce l'incertezza nelle classi considerando ogni classe come una distribuzione di probabilità discreta. Questo approccio si traduce in una maggiore robustezza in scenari con classi rumorose o dataset provenienti da diverse strumentazioni.

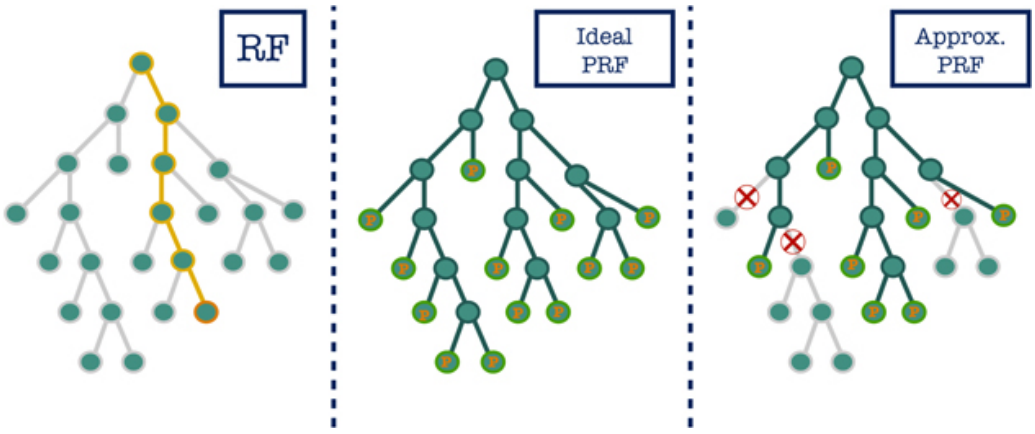


Figura 3.6: Illustrazione della propagazione di un oggetto in un albero RF, in un PRF ideale e in un PRF con propagazione selettiva basata su una soglia di probabilità. Il pannello di sinistra mostra la propagazione di un singolo oggetto in un albero RF tradizionale, in cui l'oggetto segue un'unica traiettoria (evidenziata nella figura) determinata dalle soglie binarie nei nodi di decisione. Il pannello centrale rappresenta la propagazione dello stesso oggetto in un albero PRF ideale, dove l'oggetto si propaga lungo tutti i rami, con le probabilità associate calcolate e registrate a ogni biforcazione. Infine, il pannello di destra mostra la propagazione dell'oggetto in un PRF con soglia di probabilità. In questo caso, i rami con una probabilità inferiore alla soglia prefissata vengono scartati, come indicato dalle X nella figura. Questo meccanismo permette di considerare solo i rami con probabilità significativa, riducendo il tempo di esecuzione senza compromettere la qualità della classificazione.

3.4.2 Funzionamento del PRF

Il funzionamento del PRF può essere descritto nei seguenti passaggi fondamentali, che lo differenziano dal tradizionale algoritmo RF:

1. **Rappresentazione probabilistica delle feature**: L'input di un problema di classificazione supervisionata include due tipi di incertezze:

   (a) **Incertezza sulle feature** Δx, che rappresenta l'incertezza associata alla misura delle caratteristiche di un oggetto (ad esempio, errori fotometrici nelle osservazioni astronomiche).

   (b) **Incertezza sulle classi** Δy, che riflette l'incertezza nella classificazione dell'oggetto durante la fase di training.

   Nel PRF, ogni oggetto non viene rappresentato da un vettore di caratteristiche deterministico

$$\mathbf{x} = [x_1, x_2, ..., x_d]$$

bensì da un vettore di distribuzioni probabilistiche

$$\mathbf{X} = [X_1, X_2, ..., X_d]$$

in cui ogni X_j è una variabile casuale che descrive l'incertezza sulla feature j. Si assume che le feature siano distribuite normalmente:

$$X_{i,j} \sim N(\mu_{i,j}, \sigma^2_{i,j})$$

dove $X_{i,j}$ rappresenta la j-esima feature dell'i-esimo oggetto, con media $\mu_{i,j}$ e varianza $\sigma^2_{i,j}$. Tuttavia, il PRF non è limitato a distribuzioni gaussiane e può gestire qualsiasi distribuzione assegnata alle feature.

Considerando un problema di classificazione binaria tra due classi A e B, la classe assegnata all'oggetto i, C_i, è una variabile casuale di Bernoulli con probabilità associata:

$$p_{iA} = P(C_i = A) = 1 - P(C_i = B)$$

Nel caso generale multi-classe, la classificazione può essere rappresentata come una distribuzione discreta sulle classi possibili:

$$P(C_i = c) = p_{i,c}, \quad \text{con} \quad \sum_{c=1}^{C} p_{i,c} = 1$$

dove $p_{i,c}$ rappresenta la probabilità che l'oggetto i appartenga alla classe c.

Questa rappresentazione probabilistica consente al PRF di modellare sia le incertezze nelle feature di input che nelle etichette di training, migliorando la robustezza del modello in presenza di dati rumorosi o mal etichettati.

2. **Decisioni probabilistiche dei nodi**: Nei Decision Tree tradizionali, ogni nodo è associato a una condizione deterministica:

   $X_j > \chi$ (ramo destro) oppure $X_j < \chi$ (ramo sinistro)

   dove χ è una soglia.

   Nel PRF, la propagazione dell'i-esimo oggetto all'interno dell'albero avviene con una probabilità determinata dalla combinazione delle probabilità associate a ciascun nodo attraversato. Se il primo nodo si basa sulla caratteristica X_k con soglia χ_1, l'oggetto si propaga:

verso destra con probabilità

$$\pi_{i,k}(r) = F_{i,k}(\chi_1) = \int_{\chi_1}^{\infty} p(X_{i,j}) dX_{i,j}$$

verso sinistra con probabilità:

$$\pi_{i,k}(l) = 1 - F_{i,k}(\chi_1) = \int_{-\infty}^{\chi_1} p(X_{i,j}) dX_{i,j}$$

dove $p(X_{i,j})$ è la funzione di densità di probabilità della variabile $X_{i,j}$.

Se si assume che $X_{i,j}$ segua una distribuzione normale con media $\mu_{i,j}$ e varianza $\sigma^2_{i,j}$, queste probabilità possono essere esplicitate in termini della funzione di distribuzione cumulativa della normale standard:

per il ramo destro:

$$\pi_{i,j}(r) = \Phi\left(\frac{\chi_1 - \mu_{i,k}}{\sigma_{i,k}}\right)$$

per il ramo sinistro:

$$\pi_{i,j}(l) = 1 - \Phi\left(\frac{\chi_1 - \mu_{i,k}}{\sigma_{i,k}}\right)$$

dove Φ rappresenta la funzione cumulativa della distribuzione normale standard.

Ogni campione viene suddiviso fra i rami in modo proporzionale a $\pi_{i,j}(r)$ e $\pi_{i,j}(l)$, incorporando così l'incertezza delle feature nel processo di propagazione.

Per un nodo n situato a una profondità maggiore nell'albero, la probabilità che l'i-esimo oggetto lo raggiunga è data dalla combinazione delle biforcazioni precedenti.

Ad esempio, se un oggetto ha seguito due volte la direzione destra e poi una volta la sinistra ($n = r, r, l$), la probabilità complessiva è:

$$\pi_i(r, r, l) = F_{i,k_1}(\chi_1) \times F_{i,k_2}(\chi_2) \times (1 - F_{i,k_3}(\chi_3))$$

Generalizzando ad un nodo arbitrario si ha:

$$\pi_i(n) = \prod_{\eta \in R} F_{i,k_\eta}(\chi_\eta) \times \prod_{\xi \in L} (1 - F_{i,k_\xi}(\chi_\xi))$$

3. **Propagazione selettiva con soglia di probabilità**: Propagare tutti gli oggetti in tutti i rami dell'albero può essere computazionalmente costoso. Per ottimizzare l'efficienza, si introduce una soglia di probabilità p_{th} che stabilisce un limite inferiore alla propagazione: $\pi(n) > p_{th}$ affinché un oggetto possa propagarsi fino al nodo n (Figura 3.6). Questo approccio riduce il numero di calcoli mantenendo inalterata la qualità delle predizioni.

4. **Aggregazione nei nodi foglia**: Nei nodi foglia, gli oggetti vengono rappresentati tramite una distribuzione di probabilità sulle classi. Per ogni classe c, il PRF calcola la probabilità attesa $\overline{\Pr}_{n,c}$ considerando le distribuzioni degli oggetti propagati:

$$\overline{P}_{n,c} = \frac{\sum_{i \in n} \pi_i(n) p_{i,c}}{\sum_{i \in n} \pi_i(n)}$$

5. **Predizione probabilistica**: La classe finale viene determinata massimizzando la probabilità aggregata sui diversi alberi della foresta:

$$\hat{c} = \arg\max_{c} \Pr(c)$$

Questa rappresentazione viene integrata nella fase di addestramento, migliorando la robustezza del modello.

Grazie alla sua struttura probabilistica, il **PRF** è particolarmente robusto nei seguenti scenari:

- **dati rumorosi**: gestisce incertezze nelle caratteristiche di input e nelle classi;
- **dati non bilanciati**: le predizioni probabilistiche riducono il rischio di bias verso le classi maggioritarie;
- **domain adaptation**: adatta meglio le predizioni in presenza di differenze statistiche tra il dataset di training e quello di test.

3.4.3 Vantaggi del PRF

Il **PRF** ha mostrato un miglioramento significativo rispetto al tradizionale **RF**, soprattutto in contesti in cui le incertezze nei dati sono marcate e influenzano la qualità delle predizioni. Grazie alla sua capacità di trattare le incertezze come distribuzioni probabilistiche, il PRF è in grado di sfruttare al meglio le informazioni incomplete o inaffidabili, migliorando l'accuratezza della classificazione rispetto al RF, che considera i dati come deterministici. Sebbene l'implementazione del PRF comporti tempi di esecuzione più lunghi, il miglioramento delle performance predittive giustifica l'adozione del PRF in scenari complessi, come quelli astronomici, dove le incertezze nelle caratteristiche e nelle classi sono una costante. Questo approccio potrebbe portare a una gestione efficace di dataset caratterizzati da rumore elevato o distribuzioni incerte, aprendo la strada a nuove applicazioni in ambiti dove l'affidabilità dei dati è un aspetto cruciale.

Nel capitolo successivo, esploreremo nel dettaglio come ciascuno di questi algoritmi di classificazione è stato implementato e applicato al nostro dataset, descrivendo le scelte metodologiche,

la gestione dei dati e le tecniche adottate per ottimizzare le prestazioni del modello. Vengono inoltre analizzate le specifiche implementazioni utilizzate, includendo i parametri scelti e le operazioni di preprocessing effettuate.

Capitolo 4

Implementazione degli Algoritmi di Classificazione

Questo capitolo descrive in dettaglio la costruzione e la valutazione dei modelli di classificazione impiegati nell'analisi. L'obiettivo è fornire una panoramica completa delle metodologie adottate, con particolare attenzione all'implementazione degli algoritmi di classificazione e alla valutazione delle loro prestazioni.

In primo luogo, vengono descritte in dettaglio le operazioni di pre-processing applicate ai dati prima della fase di classificazione, con particolare attenzione alla selezione delle caratteristiche e al bilanciamento delle classi. Un aspetto cruciale riguarda la gestione delle incertezze nei dati osservativi, affrontata attraverso l'implementazione del metodo Monte Carlo per il modello Random Forest e per la sua combinazione con LDA. L'algoritmo PRF, invece, incorpora nativamente la gestione delle incertezze, rendendo superflua l'applicazione di tale metodo.

Successivamente, vengono illustrate le strategie di valutazione delle prestazioni, basate su metriche standard quali accuratezza, precisione, recall e F1-score, con particolare attenzione al confronto tra i diversi approcci su validation set e test set indipendenti.

Si passa poi alla presentazione dei modelli di classificazione adottati, tra cui Random Forest (RF), RF combinato con LDA e PRF. Per ciascun modello vengono descritte le tecniche di addestramento e ottimizzazione adottate, evidenziando le scelte effettuate per adattarli alle specificità dei dati analizzati.

Infine, vengono discussi i risultati ottenuti, mettendo in evidenza le differenze tra i modelli in termini di accuratezza e robustezza. L'analisi consente di identificare il modello più performante e di valutare l'impatto delle diverse strategie di gestione delle incertezze e di bilanciamento delle classi.

4.1 Preprocessing dei dati

Nella fase di preprocessing dei dati, sono state eseguite diverse operazioni cruciali per preparare il dataset alla fase di modellazione. Per questa fase è stata utilizzata la funzione `preprocessing` (vedere Appendice C), che si occupa di gestire vari aspetti del dataset, come la rimozione di oggetti non rilevanti, il calcolo dei colori fotometrici e la gestione delle incertezze tramite il metodo Monte Carlo. Sebbene la funzione `preprocessing` non includa la suddivisione del dataset in training, validation e test set, né il bilanciamento del dataset, queste operazioni fanno comunque parte del processo complessivo di preparazione del dataset.

Nei seguenti sottoparagrafi, verranno descritti in dettaglio i passaggi principali eseguiti durante il preprocessing, tra cui l'eliminazione degli oggetti non rilevanti, il calcolo e l'analisi dei colori, la suddivisione del dataset e il bilanciamento delle classi.
La gestione delle incertezze, che avviene separatamente tramite il metodo Monte Carlo è integrata nel flusso di preprocessing, e in questa fase vengono generati campioni sintetici che considerano le incertezze sulle magnitudini e i colori, come descritto nel paragrafo dedicato.

4.1.1 Eliminazione degli oggetti non rilevanti

Durante la fase di preprocessing, è stato necessario selezionare solo gli oggetti pertinenti per l'analisi al fine di migliorare l'efficacia del modello. Pertanto, nel dataset di SHELLQs sono stati rimossi gli oggetti con etichetta `OIII` ([OIII] emitters), in quanto non rilevanti per gli scopi della tesi. Questi oggetti non appartenevano alle classi di interesse per la nostra analisi e sono stati esclusi durante l'addestramento di tutti i modelli.

Per l'addestramento dei modelli Random Forest combinato con LDA e Probabilistic Random Forest, oltre alle [OIII] emitters, sono stati eliminati anche i **quasar** dal dataset di SHELLQs. Questo perché il dataset SHELLQs è stato utilizzato come training e validation set, mentre il dataset J1030 è stato impiegato come test set indipendente. Dato che il dataset J1030 include solo **galassie** e **brown dwarf**, si è scelto di rendere coerente la composizione del dataset di training con quella del test set, rimuovendo i quasar per evitare incongruenze tra i due domini.

Questa scelta è stata fondamentale per garantire che il modello fosse adeguatamente addestrato e validato sulle stesse classi presenti nel test set, consentendo una valutazione più accurata delle sue capacità di generalizzazione.

4.1.2 Divisione dei dataset

La corretta suddivisione del dataset è una fase fondamentale per garantire una valutazione accurata delle prestazioni dei modelli. A seconda del metodo di classificazione utilizzato, sono state adottate diverse strategie di divisione del dataset di SHELLQs.

Nella fase iniziale del progetto, in cui non sono state ancora prese in considerazione le incertezze e il dataset J1030 non è stato utilizzato come test set, sono stati creati tre insiemi distinti per addestrare e valutare il modello Random Forest:

- **Training set**: utilizzato per addestrare il modello.
- **Validation set**: utilizzato per ottimizzare i parametri e prevenire l'overfitting.
- **Test set**: utilizzato per valutare le prestazioni finali del modello.

In questa fase, l'intero processo di addestramento e valutazione è stato effettuato esclusivamente utilizzando il dataset SHELLQs.

Questa suddivisione è stata effettuata utilizzando la funzione `train_test_split` [1] di `scikit-learn`, con una proporzione standard di 80% per il training set, 10% per il validation set e 10% per il test set.

Successivamente, il dataset J1030 è stato utilizzato come test set indipendente, consentendo una valutazione più realistica delle capacità di generalizzazione del modello su un dominio diverso da quello di SHELLQs e più simile a quello dei dati che l'algoritmo dovrà classificare in futuro. La scelta di J1030 come test set si è rivelata cruciale, poiché presenta una distribuzione delle caratteristiche e una composizione delle classi diversa rispetto a SHELLQs, rendendolo un banco di prova significativo per la robustezza dei modelli. In questa fase, la suddivisione del dataset SHELLQs è stata semplificata in due insiemi: training set e validation set, garantendo che ogni configurazione avesse a disposizione dati adeguati per il training e la validazione, mantenendo al contempo un test set indipendente per una valutazione affidabile e imparziale.

4.1.3 Bilanciamento del dataset

Un aspetto cruciale della preparazione del dataset è stato affrontare il problema del bilanciamento delle classi, dato che il dataset di SHELLQs presenta una distribuzione fortemente sbilanciata tra Quasar (162 oggetti), Brown Dwarf (96 oggetti) e Galassie (38 oggetti). Il bilanciamento del dataset è fondamentale, poiché un dataset sbilanciato può portare a modelli che predicono in modo errato le classi meno rappresentate. Infatti, se una classe è significativamente più numerosa delle altre, il modello potrebbe imparare a predire principalmente la classe maggioritaria, trascurando

[1] Per maggiori dettagli, visitare `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`.

quelle minoritarie. Questo compromette la capacità del modello di generalizzare correttamente, in particolare quando deve fare previsioni su nuovi dati. Bilanciare il dataset garantisce che tutte le classi siano equamente rappresentate durante l'addestramento, migliorando così le performance del modello e assicurando che apprenda caratteristiche significative da ciascuna classe.

Per questo motivo, sono stati testati due approcci: l'addestramento del modello Random Forest (RF) senza alcun tipo di resampling e l'addestramento su un training set bilanciato utilizzando `SMOTE` (Synthetic Minority Over-sampling Technique)[2]. Questo confronto ha permesso di valutare se il bilanciamento migliorasse la capacità di classificazione, in particolare per le classi meno rappresentate.

`SMOTE` è stato applicato esclusivamente al training set, lasciando invariati il validation set e il test set. Questo approccio è stato adottato per evitare di alterare la distribuzione delle classi nei dati di test e di validazione, preservando così l'integrità del processo di valutazione del modello. Il metodo SMOTE genera nuovi campioni sintetici per le classi minoritarie attraverso un processo di interpolazione. In pratica, per ogni campione appartenente a una classe minoritaria, SMOTE seleziona i k campioni più simili nello spazio delle caratteristiche e crea nuovi campioni sintetici posizionandoli casualmente lungo il segmento che collega il campione originale a uno dei suoi vicini. Questo approccio migliora la rappresentatività delle classi minoritarie senza duplicare semplicemente i dati esistenti, riducendo così il rischio di overfitting.

Dopo l'applicazione di SMOTE al training set, tutte le classi sono state portate a un numero equivalente di campioni. Questa configurazione è stata utilizzata per addestrare il modello RF, consentendo un confronto diretto tra le prestazioni del modello su training set bilanciati e sbilanciati.

Il confronto ha evidenziato l'importanza del bilanciamento dei dati nei contesti di classificazione multi-classe. In particolare, SMOTE ha mostrato di poter migliorare l'accuratezza complessiva e la capacità del modello di classificare correttamente le classi meno rappresentate, riducendo l'effetto dello sbilanciamento nel dataset di training.

4.2 Gestione delle incertezze nei dati

Nel contesto dell'analisi di dati astronomici, è fondamentale tenere conto delle incertezze associate alle misurazioni delle magnitudini e dei colori, poiché queste misurazioni possono essere soggette a errori legati agli strumenti di osservazione, alla qualità dei dati e ad altri fattori. Per integrare queste incertezze nel modello di classificazione, sono stati adottati due approcci

[2]Per maggiori dettagli, visitare `https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html`.

distinti: il metodo Monte Carlo e il Probabilistic Random Forest.

Il metodo Monte Carlo è stato applicato esclusivamente nel caso dei modelli Random Forest e Random Forest combinato ad LDA, al fine di generare campioni sintetici delle caratteristiche, tenendo conto degli errori relativi associati a ciascuna magnitudine. Questo approccio consente di simulare i campioni sintetici, migliorando la rappresentazione delle incertezze nei dati. La gestione delle incertezze tramite Monte Carlo è stata implementata attraverso la funzione `generate_samples` (vedere Appendice C), che applica il metodo per simulare i campioni sintetici in base agli errori di misura.

Un altro approccio utilizzato è stato il Probabilistic Random Forest, che, a differenza del Monte Carlo, include direttamente la gestione delle incertezze nel processo di classificazione. Questo modello tiene conto delle incertezze durante l'addestramento e la previsione, senza necessità di un ulteriore passaggio di simulazione tramite Monte Carlo.

L'inclusione delle incertezze in entrambi i casi permette di ottenere una rappresentazione più realistica e robusta dei dati, migliorando la capacità predittiva del modello, soprattutto in presenza di rumore e imprecisione.

Nelle sezioni successive, oltre al metodo Monte Carlo, vengono descritti anche i due metodi principali utilizzati per trattare le magnitudini non rilevate: l'applicazione di una distribuzione gaussiana troncata e la stima delle magnitudini non rilevate.

4.2.1 Metodo Monte Carlo

Il metodo Monte Carlo rappresenta uno strumento fondamentale per incorporare in modo sistematico le incertezze nei dati nei processi di analisi e classificazione. Questa tecnica sfrutta simulazioni probabilistiche per generare una popolazione sintetica di dati, rappresentativa delle caratteristiche osservate e delle loro incertezze, attraverso campionamenti casuali basati su distribuzioni probabilistiche.

Nel contesto di questo lavoro, il metodo Monte Carlo è stato utilizzato per trattare in modo rigoroso le incertezze associate alle magnitudini osservate. L'obiettivo principale è quello di creare un dataset che rifletta non solo i valori osservati, ma anche la "variabilità" intrinseca dovuta agli errori di misura e ai limiti strumentali.

Per ciascun oggetto e per ogni magnitudine misurata, i valori sintetici sono stati generati seguendo una distribuzione normale, definita da due parametri principali:

- **Media (μ)**: corrispondente alla magnitudine osservata.

- **Deviazione standard (σ)**: determinata dall'errore relativo associato alla magnitudine.

Questa distribuzione rappresenta l'incertezza associata alla misura, riflettendo la probabilità che il valore reale della magnitudine si discosti da quello osservato entro i limiti definiti dall'errore. Per esempio, una magnitudine $i = 24.77 \pm 0.05$ viene descritta da una distribuzione normale centrata su 24.77 con deviazione standard 0.05, e ogni campione generato rappresenta una possibile realizzazione della misura.

Il metodo Monte Carlo consente di trasformare un dataset originale in una popolazione sintetica, che incorpora le incertezze in maniera esplicita. Ogni caratteristica non è più rappresentata da un unico valore osservato, ma da una distribuzione di possibili valori. Questa popolazione è stata utilizzata sia durante la fase di addestramento del modello, che durante la fase di test, garantendo una robustezza maggiore ai processi di classificazione. Nei test, ogni campione del dataset sintetico è stato classificato, e la predizione finale è stata ottenuta combinando statisticamente i risultati (ad esempio, calcolando la media o la moda delle predizioni).

L'utilizzo del metodo Monte Carlo presenta numerosi vantaggi:

- **Incorporazione delle incertezze**: integra le variazioni dei dati nei modelli, riducendo la possibilità di sottostimare l'impatto degli errori di misura.
- **Robustezza del modello**: i modelli di classificazione risultano meno sensibili agli errori osservativi, migliorando la loro capacità di generalizzare su nuovi dati.

In sintesi, l'applicazione del metodo Monte Carlo si è dimostrata un elemento cruciale per la gestione delle incertezze nei dati, fornendo un quadro più realistico delle proprietà statistiche del dataset, migliorando la capacità del modello di adattarsi a dati complessi e variabili.

Nei sottoparagrafi successivi andremo a descrivere la gestione dei valori non rilevati, utilizzando approcci differenti per stimare e integrare questi valori mancanti nel dataset.

4.2.2 Uso del valore limite come media per la distribuzione troncata

Il primo metodo adottato per gestire le magnitudini con limiti superiori e/o inferiori, ovvero valori censurati, prevede l'uso di una distribuzione gaussiana troncata. In questo approccio, invece di generare campioni da una distribuzione normale completa, i valori sintetici vengono generati all'interno di un intervallo fisico predefinito, corrispondente al limite superiore o inferiore delle magnitudini. Questo metodo consente di rispettare i vincoli fisici imposti dai dati, garantendo che i valori generati rimangano coerenti con le osservazioni disponibili.

Una distribuzione gaussiana troncata è una distribuzione normale modificata in modo che tutti i valori che escono da un certo intervallo siano esclusi. Matematicamente, la funzione di

densità di probabilità (PDF) di una distribuzione normale troncata si esprime come:

$$f(x) = \frac{\frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}{P(X_{min} \leq X \leq X_{max})}$$

dove:

- μ è la media della distribuzione,
- σ è la deviazione standard della distribuzione normale,
- X_{min} e X_{max} sono i limiti inferiori e superiori impostati dai dati censurati,
- $P(X_{min} \leq X \leq X_{max})$ è la probabilità di osservare un valore all'interno dell'intervallo di validità, che normalizza la funzione di densità per garantire che la somma delle probabilità sia 1.

Nel contesto delle magnitudini non rilevate, se ad esempio un oggetto ha un limite inferiore $i > 26$, la distribuzione della magnitudine i sarà centrata su 26 (il valore limite) e troncata da quel valore in giù (Figura 4.1). La distribuzione troncata quindi non può generare valori al di sotto di questa soglia, ma genera comunque campioni all'interno dell'intervallo definito.

La deviazione standard per la distribuzione troncata viene calcolata utilizzando la media degli errori associati alle magnitudini osservate per quella classe. Questa media degli errori fornisce una misura della dispersione dei dati e viene utilizzata come deviazione standard per generare la distribuzione gaussiana. La distribuzione risultante è centrata attorno al valore limite, riflettendo così le incertezze legate alle misurazioni.

La troncatura implica che, se il valore reale della magnitudine fosse maggiore o minore di un determinato limite, il campione sintetico generato dalla distribuzione troncata non può mai violare quel limite. Questo approccio ha il vantaggio di rispettare i vincoli fisici dei dati osservativi, evitando di generare valori che sarebbero fuori dall'intervallo di osservazione (Figura 4.1).

Sebbene questa tecnica permetta di integrare l'incertezza derivante dalle misure, va sottolineato che il metodo della distribuzione gaussiana troncata è un'approssimazione. Infatti, la troncatura non rispecchia esattamente la reale distribuzione dei valori oltre il limite osservato, ma fornisce una stima utile per l'inclusione dei dati censurati nel modello di classificazione. La troncatura implica che non possiamo generare campioni al di fuori dall'intervallo osservato, ma questo potrebbe non riflettere accuratamente la distribuzione dei valori reali, in quanto non abbiamo informazioni dirette sui dati oltre il limite.

Questa approssimazione, sebbene utile per integrare i dati censurati nel modello di classificazione, può introdurre dei bias, in quanto la distribuzione reale dei valori censurati potrebbe differire da quella assunta dalla distribuzione normale troncata. Tuttavia, nella pratica, questa

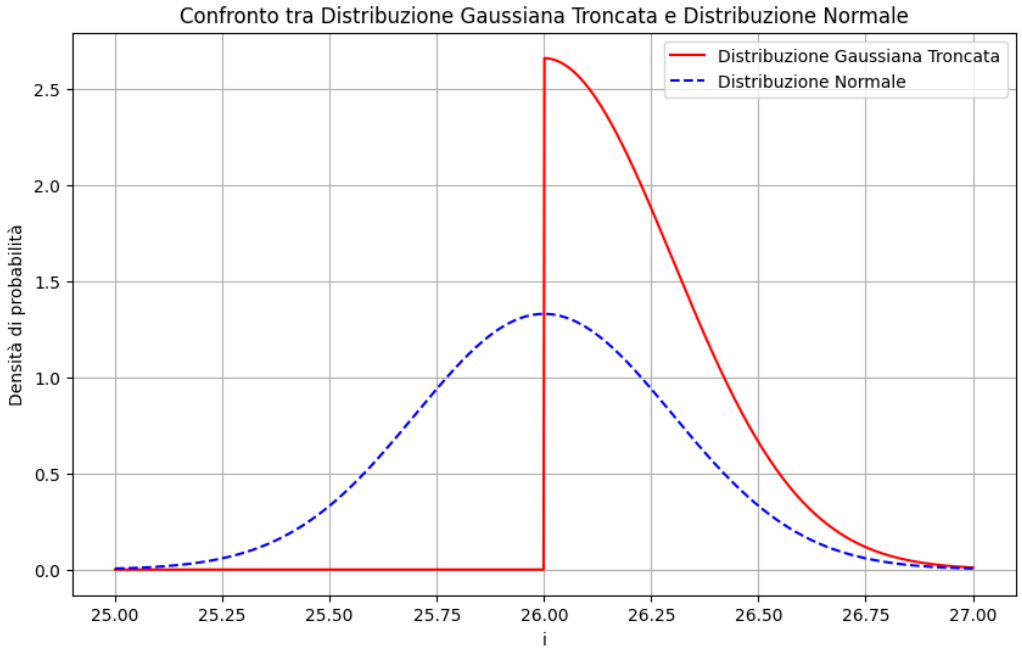


Figura 4.1: Confronto tra la distribuzione gaussiana normale (linea tratteggiata in blu) e la distribuzione gaussiana troncata (linea continua in rosso), entrambe centrate su $i = 26$. La distribuzione troncata mostra valori limitati dal limite inferiore di $i = 26$, mentre la distribuzione normale è definita su tutta la gamma di valori.

soluzione è comunemente utilizzata per trattare dati censurati, specialmente quando non è possibile ottenere stime precise per i valori mancanti o censurati.

4.2.3 Stima della magnitudine tramite colori come media della distribuzione troncata

Il secondo metodo adottato per la stima delle magnitudini in presenza di valori non rilevati in una specifica banda si basa sull'utilizzo dei colori medi delle classi per calcolare la magnitudine attesa. Ad esempio, per stimare la magnitudine i, si è utilizzato il colore medio $i - z$ calcolato per ciascuna classe. Questo processo si basa sull'assunzione che il colore medio delle classi sia rappresentativo degli oggetti non rilevati, fornendo una base ragionevole per stimare le magnitudini mancanti. La magnitudine stimata, $i_{stimata}$, è data dalla relazione:

$$i_{stimata} = z + (i - z)_{medio},$$

$(i - z)_{medio}$ è il valore medio della classe, mentre i è il valore limite della magnitudine rilevata in quella banda.

Una volta stimata la magnitudine attesa, si procede a determinare l'incertezza associata, rappresentata dalla deviazione standard a 1σ di una distribuzione normale. Per oggetti non rilevati, questa viene calcolata partendo dal limite inferiore della magnitudine, espresso nei dati

osservativi come un valore limite a $X\sigma$ (ad esempio, $i > 26$ corrisponde al limite inferiore a 2σ), dove X è 5 per i dati SHELLQs e 2 per i dati J1030. In questo caso, il valore di 1σ è determinato dalla relazione:

$$1\sigma = |\frac{i_{lower\ limits} - i_{stimata}}{X}|$$

Con il valore stimato $i_{stimato}$ come media e 1σ calcolato come descritto sopra, si genera una distribuzione normale semplice per rappresentare la magnitudine. La distribuzione normale è definita come:

$$p(i) = N(i_{stimato}, \sigma^2)$$

dove:

- $i_{stimato}$ è il valore stimato della magnitudine,
- σ è la deviazione standard calcolata dal limite inferiore.

Campioni sintetici vengono quindi generati da questa distribuzione normale e utilizzati nel metodo Monte Carlo per integrare le incertezze nel modello di classificazione.

Supponiamo per esempio di avere per un oggetto un lower limit a 2σ in banda i di 25.92, una magnitudine in banda z di 23.73 e che il valore medio di $i-z$ sia 2.97 per la classe corrispondente. La magnitudine stimata in i sarà dunque $i_{stimata} = 26.70$ e il valore di 1σ sarà di 0.39. La magnitudine finale verrà campionata da una distribuzione normale con media $\mu = 26.70$ e deviazione standard $\sigma = 0.39$ (Figura 4.2).

Questo approccio non fornisce il valore effettivo della magnitudine non rilevata, ma una stima basata sui dati disponibili e su ipotesi statistiche ragionevoli. Sebbene tale metodo non sia rigoroso, è efficace nel rappresentare le incertezze associate ai dati censurati. Rispetto a una distribuzione troncata, il metodo appena descritto semplifica la simulazione senza compromettere significativamente la robustezza del modello. Pur essendo una stima approssimativa che potrebbe non riflettere pienamente la distribuzione reale delle magnitudini oltre i limiti osservativi, tale metodo offre comunque una rappresentazione più realistica della variabilità effettiva dei dati, evitando di limitare artificialmente i valori generati entro un intervallo ristretto. Questo approccio integra in modo sistematico le incertezze dei dati censurati nel modello predittivo, migliorando la sua capacità di generalizzazione e la robustezza nelle fasi di classificazione.

4.3 Valutazione delle prestazioni

La valutazione delle prestazioni dei modelli è una fase cruciale per determinare la loro capacità di generalizzare su nuovi dati. In questo lavoro, sono state adottate metriche standard di classificazione per analizzare l'accuratezza e l'affidabilità dei modelli sviluppati. Questa fase ha

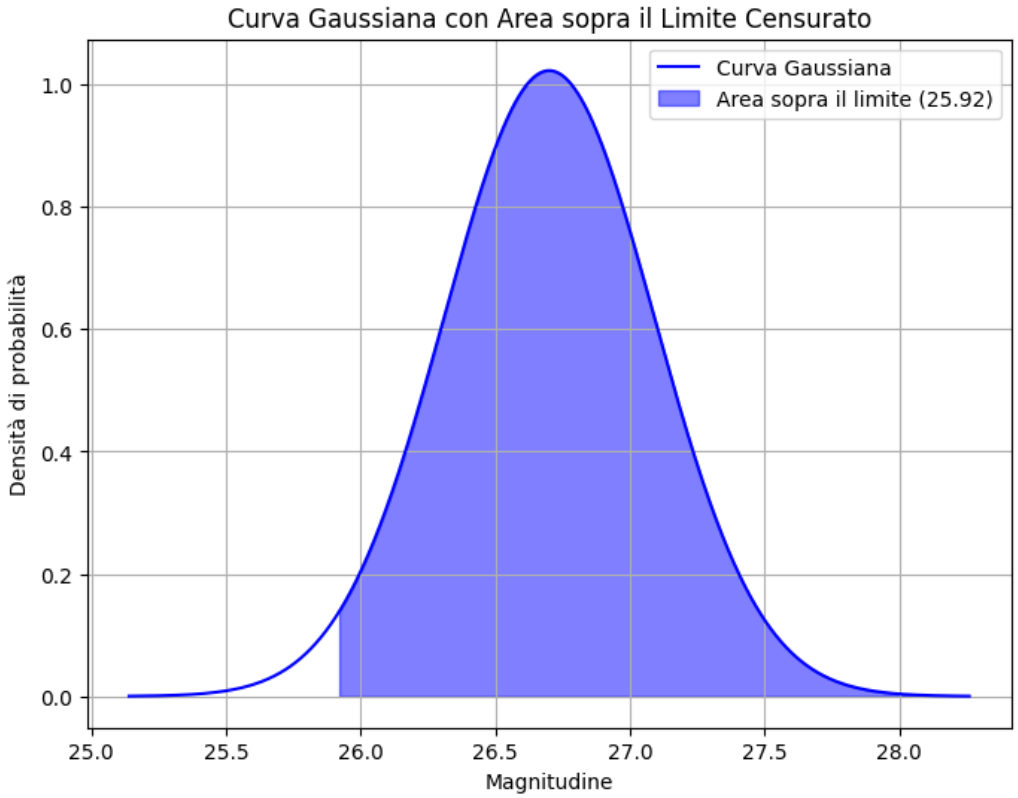


Figura 4.2: Distribuzione normale con media $\mu = 26.70$ e deviazione standard $\sigma = 0.39$. L'area colorata in blu rappresenta i possibili valori della magnitudine stimata al di sopra del limite inferiore di 25.92.

permesso non solo di confrontare le prestazioni tra i diversi algoritmi implementati, ma anche di identificare eventuali criticità, come la sovrapposizione tra classi o l'effetto dello sbilanciamento nei dataset.

Per automatizzare e uniformare il processo, è stata sviluppata la funzione `evaluate` (vedere Appendice C), che consente di calcolare metriche fondamentali, visualizzare la matrice di confusione e salvare i risultati per un'analisi successiva. Di seguito viene descritta l'applicazione di questa funzione sui validation set e test set.

4.3.1 Metriche di valutazione (Accuracy, Precision, Recall, F1-score)

Per misurare l'efficacia dei modelli di classificazione, è essenziale utilizzare metriche che forniscano una valutazione completa delle loro prestazioni. Ogni metrica fornisce un punto di vista diverso sulla capacità del modello di distinguere correttamente tra le classi, considerando sia gli errori di classificazione sia la distribuzione delle classi. Un ottimo strumento per visualizzare questi errori è la **matrice di confusione**, che mostra il numero di predizioni corrette e errate per ciascuna classe del modello. L'asse x rappresenta le classi predette, mentre l'asse y indica le classi reali (Figura 4.3).

La funzione `evaluate` è stata progettata per calcolare le metriche più comuni e rilevanti per problemi di classificazione multi-classe: **Accuracy**, **Precision**, **Recall** e **F1-score**. Queste metriche sono state selezionate per garantire una valutazione bilanciata, considerando sia le prestazioni complessive del modello sia quelle specifiche per ciascuna classe.

Di seguito vengono descritte le metriche calcolate e il loro significato nell'ambito di questa analisi.

- **Accuracy**: Misura la proporzione di predizioni corrette rispetto al totale delle osservazioni. È definita come:
$$Accuracy = \frac{Predizioni\ corrette}{Totale\ oggetti}$$
Sebbene sia una metrica intuitiva, l'accuracy può essere fuorviante in presenza di dataset sbilanciati, dove le classi maggioritarie influenzano maggiormente il risultato. In una matrice di confusione, le predizioni corrette sono rappresentate dalla somma delle diagonali (cioè, i veri positivi per ciascuna classe). Per esempio l'accuracy della matrice di confusione in Figura 4.3 è
$$Accuracy = \frac{10 + 17}{10 + 17 + 5 + 2} = \frac{27}{34} \approx 0.79 \tag{4.1}$$

- **Precision**: Valuta la frazione di predizioni corrette tra quelle appartenenti a una determinata classe predetta. È definita come:
$$Precision = \frac{Veri\ Positivi(TP)}{Veri\ Positivi(TP) + Falsi\ Positivi(FP)}$$
Questa metrica è particolarmente utile quando si desidera ridurre i falsi positivi. Una precision alta indica che il modello è accurato nel classificare una classe specifica.
Dalla matrice di confusione in Figura 4.3 abbiamo:
$$Precision = \frac{10}{10 + 5} \approx 0.67 \tag{4.2}$$

- **Recall (o sensibilità)**: misura la capacità del modello di identificare correttamente tutti i campioni appartenenti a una determinata classe, evidenziando l'efficacia nel ridurre i falsi negativi. È definita come:
$$Recall = \frac{Veri\ Positivi(TP)}{Veri\ Positivi(TP) + Falsi\ Negativi(FN)}$$
Una recall elevata indica che il modello identifica correttamente la maggior parte degli esempi positivi di una classe.
Dalla matrice di confusione in Figura 4.3 abbiamo:
$$Recall = \frac{10}{10 + 2} \approx 0.83 \tag{4.3}$$

- **F1-score**: rappresenta la media armonica di precision e recall, fornendo un compromesso tra le due metriche. È utile quando le classi sono sbilanciate e si desidera un equilibrio tra falsi positivi e falsi negativi, è definito come:

$$F1 - score = 2\frac{Precision \times Recall}{Precision + Recall}$$

Valori elevati di F1-score indicano che il modello gestisce bene il compromesso tra precision e recall.
Dalla matrice di confusione in Figura 4.3 abbiamo:

$$F1 - score = 2\frac{0.67 \times 0.83}{0.67 + 0.83} \approx 0.74 \tag{4.4}$$

Queste metriche sono calcolate per ciascuna classe e in forma aggregata, consentendo un'analisi dettagliata delle prestazioni del modello sia a livello globale sia per ciascuna classe.

4.3.2 Valutazione su diversi dataset

Per garantire una valutazione completa delle prestazioni, i modelli sono stati testati e valutati con le metriche appena descritte sul *validation set* e sul *test set*. Questo approccio consente di analizzare non solo le capacità di apprendimento dei modelli, ma anche la loro generalizzazione su dati mai visti prima.

La funzione `evaluate` segue un flusso standard per ogni set di dati analizzato:

1. **Salvataggio del modello**: Ogni modello addestrato viene salvato in formato pickle per consentirne un futuro riutilizzo o un'analisi post-addestramento.
2. **Predizioni e calcolo dell'accuratezza**: Il modello genera predizioni sui dati forniti, e l'accuratezza viene calcolata come proporzione di predizioni corrette rispetto al totale.
3. **Matrice di confusione**: Per ogni set di test o validation, viene generata una matrice di confusione (Figura 4.3).
4. **Report dettagliato delle metriche**: La funzione calcola le metriche descritte sopra (Accuracy, Precision, Recall e F1-score) per ogni classe, e salva un report in formato CSV per una successiva analisi.

Questo processo è stato applicato:

- **Sul validation set**, che rappresenta una porzione del training set di SHELLQs utilizzata per ottimizzare i parametri del modello e per monitorarne le prestazioni durante l'addestramento.

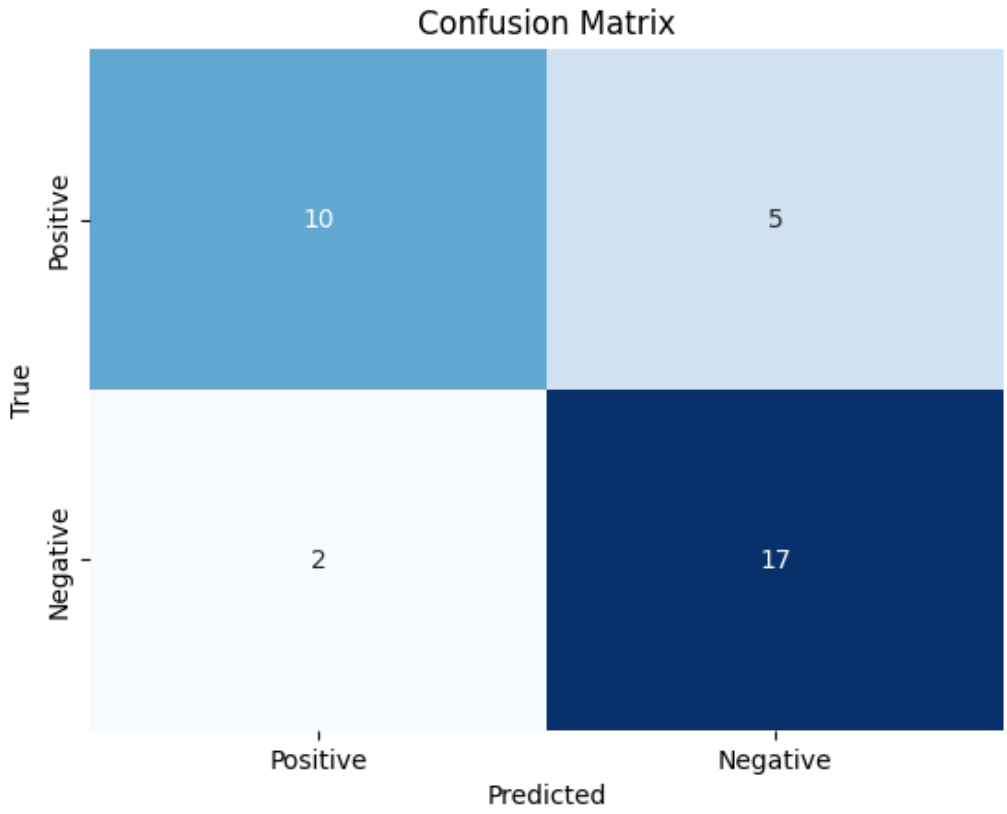


Figura 4.3: La figura rappresenta un esempio di **matrice di confusione** per un modello di classificazione binaria con le classi "Positive" e "Negative". La matrice mostra che il modello ha classificato correttamente 10 campioni come "Positive" (True Positive) e 17 campioni come "Negative" (True Negative). 2 campioni sono stati classificati erroneamente come "Positive" pur appartenendo alla classe "Negative" (False Positive) e 5 campioni, appartenenti alla classe "Positive", sono stati classificati come "Negative" (False Negative). Questi risultati evidenziano le prestazioni del modello, permettendo di calcolare metriche fondamentali come l'accuratezza, la precisione, la recall e l'F1-score.

- **Sul test set derivato dal dataset SHELLQs**, quando non è stato usato un dataset indipendente. In questo caso, una parte del dataset è stata riservata esclusivamente per la fase di test, seguendo una divisione standard.
- **Sui dati indipendenti di J1030**, per valutare la capacità del modello di generalizzare a un dataset con caratteristiche e distribuzioni diverse rispetto al training set.

Grazie a questo approccio uniforme, è stato possibile confrontare le prestazioni del modello su diverse configurazioni di test, identificando punti di forza e debolezza. L'impiego delle metriche calcolate, in particolare l'F1-score e la matrice di confusione, ha consentito una valutazione approfondita dell'efficacia del modello, evidenziando le classi con prestazioni migliori e peggiori.

Nel seguito, vengono descritte le specifiche tecniche dei modelli di classificazione adottati: **Random Forest**, **Random Forest con LDA** e **Probabilistic Random Forest**. Mentre il funzionamento teorico di ciascun algoritmo è stato trattato nel capitolo precedente, nelle prossime sezioni ci concentreremo sull'adattamento di questi modelli al nostro caso d'uso, includendo la selezione degli iperparametri e le tecniche di ottimizzazione applicate.

In particolare, per ogni modello, sono stati adottati due approcci per la gestione delle magnitudini non rilevate:

- **Approccio "Lower Limit Mean" (LLM)**: utilizzo dei limiti inferiori come media della distribuzione gaussiana troncata.
- **Approccio "Color-Based Estimation" (CBE)**: utilizzo delle magnitudini stimate tramite i colori medi delle classi come media della distribuzione gaussiana troncata.

Solo il modello RF è stato testato anche utilizzando direttamente i valori nominali (**NV**) dei limiti inferiori.

4.4 Random Forest

Per implementare il modello di **Random Forest**, è stato utilizzato il framework `scikit-learn` [3].

Il modello è stato inizialmente addestrato sul dataset SHELLQs, non tenendo conto delle incertezze sui dati e utilizzando l'approccio NV, suddiviso in training, validation e test set e successivamente, è stato addestrato nuovamente sullo stesso dataset, ma con il training set bilanciato, al fine di migliorare la rappresentatività delle classi nel modello, come descritto nella sezione 4.1.3.

La configurazione del modello si è basata sui seguenti iperparametri principali:

- **Numero di alberi (`n_estimators`)**:
  Questo parametro definisce il numero di alberi da includere nella foresta. Un numero maggiore di alberi tende a migliorare la robustezza del modello riducendo il rischio di overfitting, ma può aumentare i tempi di calcolo. Nel tuning del modello, è stato testato un range di valori tra 10 e 100, con incrementi di 10, per individuare il numero ottimale di alberi.
- **Profondità massima degli alberi (`max_depth`)**: Questo parametro limita la profondità degli alberi per evitare che il modello si adatti troppo strettamente ai dati di training (overfitting). È stato testato un range di profondità da 3 a 15.

[3] Per maggiori dettagli visitare `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

- **Numero minimo di campioni per la divisione** (`min_samples_split`): Questo parametro imposta il numero minimo di campioni richiesti per suddividere un nodo. Quando il numero di campioni in un nodo è inferiore a questo valore, il nodo non verrà suddiviso ulteriormente. I valori testati in questo lavoro sono stati: 5 e 10.

- **Numero minimo di campioni per foglia** (`min_samples_leaf`): Questo parametro definisce il numero minimo di campioni necessari per formare una foglia in un albero decisionale. Impostando un valore maggiore, si riduce la probabilità di creare foglie troppo specifiche, come ad esempio nodi che contengono un solo campione, migliorando così la capacità di generalizzazione del modello. I valori testati sono stati 2 e 4. Quando un nodo raggiunge un numero di oggetti pari (o superiore, nel caso si raggiunga la purezza del nodo) al valore impostato per `min_samples_leaf`, viene creato un nodo foglia. Questo processo assicura che i nodi non siano troppo "fini" o sensibili a poche osservazioni, favorendo un modello più robusto e in grado di generalizzare meglio ai dati non visti.

Per ottimizzare questi iperparametri dell'algoritmo, è stata utilizzata la funzione personalizzata `random_grid` (vedere Appendice C), che si basa sul metodo `RandomizedSearchCV`[4] fornito dalla libreria `scikit-learn`. Questa funzione implementa una ricerca randomizzata degli iperparametri, selezionando casualmente un numero predefinito di combinazioni all'interno di una griglia di valori specificati.
Più dettagliatamente, il processo è strutturato come segue:

1. Un sottoinsieme casuale delle combinazioni di iperparametri viene selezionato dalla griglia predefinita.

2. Per ciascuna combinazione selezionata, il modello viene addestrato e valutato tramite validazione incrociata. Questo metodo consiste nel suddividere il dataset di training in k sottoinsiemi, chiamati *folds*. In ogni iterazione, uno di questi *folds* viene utilizzato come set di validazione, mentre gli altri $k-1$ *folds* vengono utilizzati per addestrare il modello. Il processo si ripete k volte, utilizzando ciascun *fold* come set di validazione una volta.
   Ad esempio, se si desidera ottimizzare i parametri `n_estimators` (10,50,100) e `max_depth` (5,10,15), `RandomizedSearchCV` esplora tutte le $3 \times 3 = 9$ combinazioni possibili di questi parametri, valutando ciascuna combinazione tramite k-fold cross-validation.

3. Viene calcolata una metrica di prestazione (ad esempio, l'accuratezza o l'F1-score) per ciascuna configurazione di iperparametri.

4. La combinazione di iperparametri che ottiene le migliori prestazioni medie sulla validazione incrociata viene selezionata come ottimale.

[4]Per maggiori dettagli visitare `https://scikit-learn.org/1.5/modules/generated/sklearn.model_selection.RandomizedSearchCV.html`.

Questo approccio ha diversi vantaggi:

- Riduce il rischio di overfitting, poiché il modello viene valutato su dati non visti durante l'addestramento.
- Fornisce una stima più robusta delle prestazioni del modello rispetto a una semplice suddivisione in training e validation set.

La configurazione specifica utilizzata in questo progetto è stata:

- Numero di iterazioni (`n_iter`): 100, per esplorare 100 combinazioni casuali di iperparametri.
- Numero di fold per la validazione incrociata (`cv`): 3, per garantire che ogni campione venga utilizzato sia per l'addestramento che per la validazione.
- Parallelizzazione (`n_jobs`): impostata a -1, per sfruttare tutti i core disponibili del processore e velocizzare il processo di tuning.

Dunque, il modello ha selezionato la combinazione di iperparametri che massimizzava le prestazioni in termini di accuratezza sulla validazione incrociata riportate in Tabella 4.1.

Successivamente, l'algoritmo è stato valutato sul validation set per analizzare le metriche di prestazione e valutare la necessità di ulteriori modifiche agli iperparametri (Tabella 4.2). Infine, il modello ottimizzato è stato testato sul test set per verificare la sua capacità di generalizzazione (Tabella 4.2).

4.4.1 Performance di RF con e senza resampling

La Tabella 4.2 mostra il confronto delle performance tra il modello Random Forest (RF) e la sua versione bilanciata con SMOTE (RF+SMOTE), considerando sia il validation set che il test set e utilizzando i valori nominali per le magnitudini non rilevate.

Dai risultati emerge come RF senza SMOTE offra complessivamente buone performance, con una precisione elevata per le classi Quasar (Q) e Galassie (G). Tuttavia, uno dei limiti principali riguarda la recall della classe Galassie (G), che risulta particolarmente basso (50% nel validation set e 40% nel test set). Questo suggerisce che il modello fatica a identificare correttamente gli oggetti appartenenti a questa categoria, probabilmente a causa di una minore rappresentatività nel dataset di training. Questo comportamento è ben visibile nelle confusion matrix riportate in Figura 4.4, dove si nota un numero significativo di Galassie (G) erroneamente classificate come altre classi.

L'accuratezza complessiva del modello RF senza SMOTE è pari all'87% sia sul validation set che sul test set, come riportato in Tabella 4.3.

Modello	max_depth	min_samples_leaf	min_samples_split	n_estimators
		NV		
RF	14	2	10	30
RF + SMOTE	11	2	5	100
		CBE		
RF	5	2	10	20
RF + SMOTE	13	4	5	10
RF + LDA	14	4	5	10
RF + LDA + SMOTE	15	4	10	30
PRF	11	-	-	40
PRF + SMOTE	5	-	-	40
		LLM		
RF	4	4	5	70
RF + SMOTE	7	4	10	40
RF + LDA	4	4	10	10
RF + LDA + SMOTE	12	2	5	10
PRF	4	-	-	60
PRF + LDA	6	-	-	70

Tabella 4.1: Parametri dei modelli Random Forest (RF) e Probabilistic RF (PRF) con diverse configurazioni.

L'introduzione di SMOTE, invece, porta a un miglior bilanciamento tra le classi, aumentando la recall per la classe Brown Dwarf (BD) (dal 90% all'88% nel validation set e dal 100% al 78% nel test set). Tuttavia, questo miglioramento non si riflette in un incremento generale delle performance, poiché si osserva un calo nella precisione per le classi BD e G, con un conseguente peggioramento dell'F1-score della classe Galassie (G). Questo comportamento suggerisce che l'oversampling sintetico, pur aiutando a rendere il modello più sensibile a classi meno rappresentate, introduce anche delle difficoltà. L'effetto di questa riduzione della precisione e del peggioramento dell'F1-score è evidenziato nelle confusion matrix in Figura 4.4, dove si osserva un incremento delle predizioni errate per le classi BD e G.

A conferma di ciò, si osserva una riduzione dell'accuratezza complessiva nel modello RF+SMOTE, che passa dall'81% nel validation set al 74% nel test set (Tabella 4.3).

	Validation Set			Test Set		
	Precision	Recall	F1-score	Precision	Recall	F1-score
RF						
BD	90%	90%	90%	82%	100%	90%
G	100%	50%	67%	100%	40%	57%
Q	84%	94%	89%	89%	94%	91%
Weighted Avg	88%	87%	86%	89%	87%	85%
RF + SMOTE						
BD	69%	90%	78%	70%	78%	74%
G	50%	25%	33%	50%	20%	29%
Q	94%	88%	91%	79%	88%	83%
Weighted Avg	80%	81%	79%	72%	74%	72%

Tabella 4.2: Confronto delle performance tra RF e RF + SMOTE con NV

Le ragioni dietro questo andamento possono essere molteplici. Un primo aspetto riguarda la possibile sovradattabilità ai dati sintetici. SMOTE genera nuovi campioni interpolando tra quelli esistenti, ma se questi campioni non rappresentano fedelmente la distribuzione reale dei dati, il modello potrebbe apprendere pattern artificiali, riducendo la capacità di generalizzazione sui dati di test. Questo effetto è particolarmente evidente nei dataset di piccole dimensioni, dove anche un numero limitato di nuovi campioni sintetici può alterare la struttura originale dei dati.

Un secondo aspetto da considerare è la distribuzione intrinseca delle classi. Se le classi sono naturalmente sbilanciate e ben separate, l'introduzione di nuovi dati sintetici potrebbe confondere il modello più che aiutarlo. Questo potrebbe spiegare il peggioramento nella classificazione delle Galassie (G), che già in condizioni standard risultavano difficili da distinguere. Inoltre, la classe Quasar (Q) è ampiamente distribuita e ricopre una vasta gamma di valori nelle feature, il che potrebbe rendere ancora più complicata la separazione tra le classi e limitare l'efficacia del bilanciamento tramite SMOTE.

Un ulteriore elemento da tenere in considerazione è la maggiore variabilità nelle feature

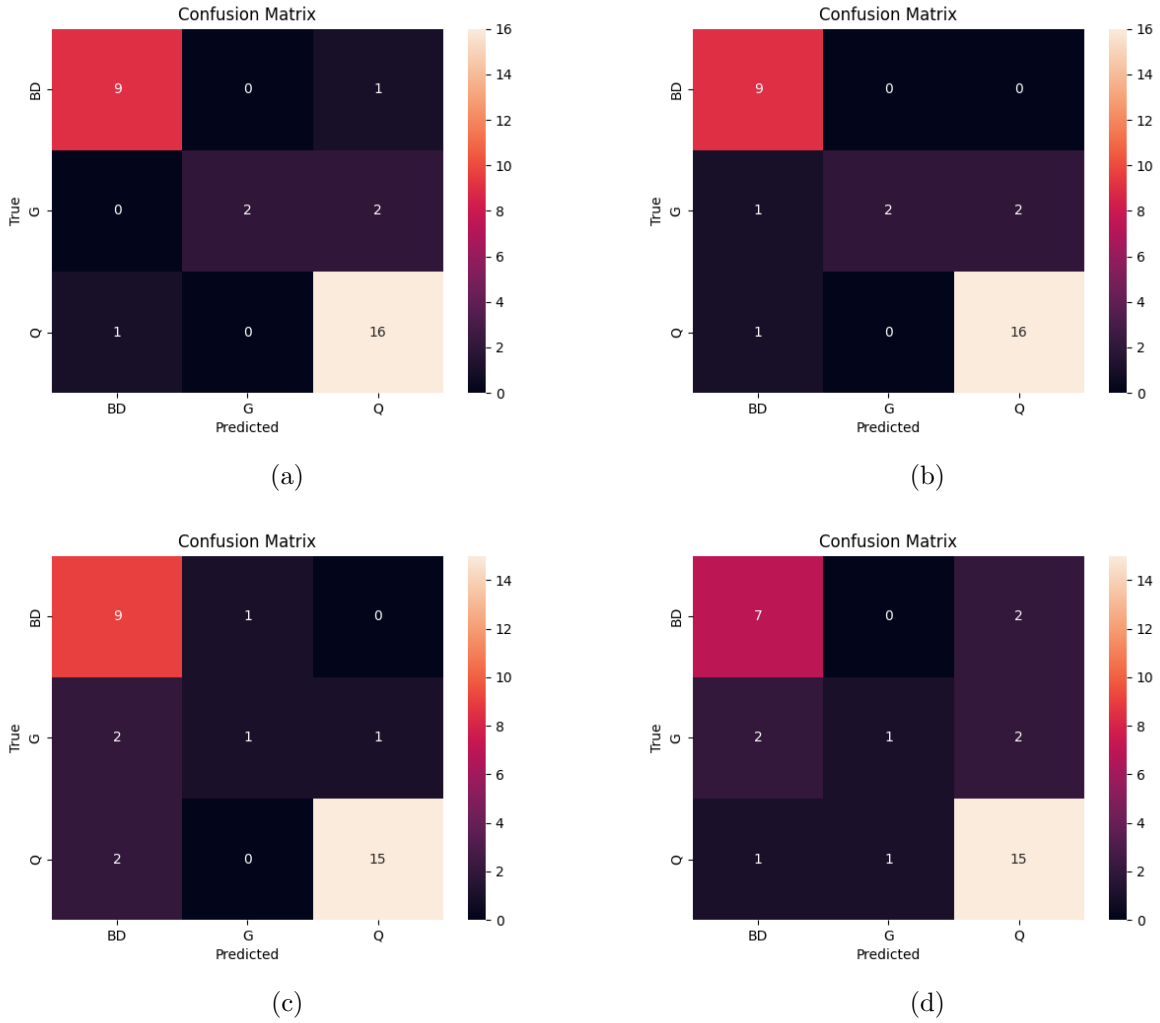
introdotta dall'oversampling sintetico. SMOTE, infatti, amplia la varianza dei dati interpolando tra punti vicini nello spazio delle feature, il che potrebbe portare a una minore stabilità nel test set, dove il modello si trova ad affrontare oggetti completamente nuovi. Questo potrebbe spiegare il calo delle performance osservato nel test set, specialmente nelle classi più difficili da classificare.

Infine, è importante sottolineare che questa trattazione non include le incertezze associate alle misure, un aspetto che potrebbe influenzare significativamente le performance del modello. In particolare, per le magnitudini non stimate sono stati utilizzati i valori nominali dei limiti inferiori, che rappresentano solo un'approssimazione poco precisa dei valori reali. Questo potrebbe aver introdotto un ulteriore elemento di distorsione nei dati, rendendo più difficile per il modello apprendere correttamente la separazione tra le classi.

Complessivamente, RF senza SMOTE ottiene migliori performance generali, mentre RF+SMOTE migliora il bilanciamento delle classi, ma con una riduzione dell'accuratezza complessiva. In particolare, l'uso di SMOTE sembra migliorare la recall per le classi meno rappresentate, ma

Accuracy		
Modello	**Validation Set**	**Test Set**
NV		
RF	87%	87%
RF + SMOTE	81%	74%
LLM		
RF	93%	91%
RF + SMOTE	88%	89%
RF + LDA	94%	93%
RF + SMOTE + LDA	96%	94%
PRF	96%	94%
PRF + SMOTE	93%	91%
CBE		
RF	89%	84%
RF + SMOTE	86%	88%
RF + LDA	93%	94%
RF + SMOTE + LDA	89%	88%
PRF	89%	88%
PRF + SMOTE	96%	97%

Tabella 4.3: Confronto delle accuracy per i diversi modelli e strategie di gestione delle magnitudini censurate.



(a) (b) (c) (d)

Figura 4.4: Matrici di confusione per il validation set ((a), (c)) e il test set ((b), (d)). I pannelli (a) e (b) corrispondono al modello Random Forest standard, mentre i pannelli (c) e (d) si riferiscono a Random Forest con SMOTE.

a discapito della precisione complessiva. Questo suggerisce che potrebbero essere necessari approcci alternativi, come tecniche di domain adaptation o modelli più robusti alla distribuzione dei dati.

4.4.2 Performance RF con Monte Carlo per la gestione delle incertezze con e senza resampling

La fase successiva è stata quella di includere le incertezze associate alle magnitudini e ai colori attraverso l'uso del metodo Monte Carlo descritto nel paragrafo 4.2.1. Questa metodologia ha richiesto l'utilizzo del dataset J1030 come test set indipendente, poiché i dati "trasformati" con Monte Carlo avrebbero potuto compromettere una valutazione imparziale. Poiché J1030 include solo oggetti appartenenti alle classi Galassie (G) e Brown Dwarf (BD), i Quasar (Q)

	Validation Set			Test Set		
	Precision	Recall	F1-score	Precision	Recall	F1-score
LLM						
RF						
BD	90%	100%	95%	84%	100%	91%
G	100%	78%	88%	100%	81%	90%
Weighted Avg	94%	93%	93%	92%	91%	91%
RF + SMOTE						
BD	94%	89%	92%	80%	100%	89%
G	80%	89%	84%	100%	75%	86%
Weighted Avg	90%	89%	89%	90%	88%	87%
CBE						
RF						
BD	86%	100%	93%	76%	100%	86%
G	100%	67%	80%	100%	69%	81%
Weighted Avg	91%	89%	89%	88%	84%	84%
RF + SMOTE						
BD	94%	84%	89%	80%	100%	89%
G	73%	89%	80%	100%	75%	86%
Weighted Avg	87%	86%	86%	90%	88%	87%

Tabella 4.4: Performance del modello Random Forest (RF) con i due diversi metodi di gestione delle magnitudini non rilevate (LLM e CBE) e resampling.

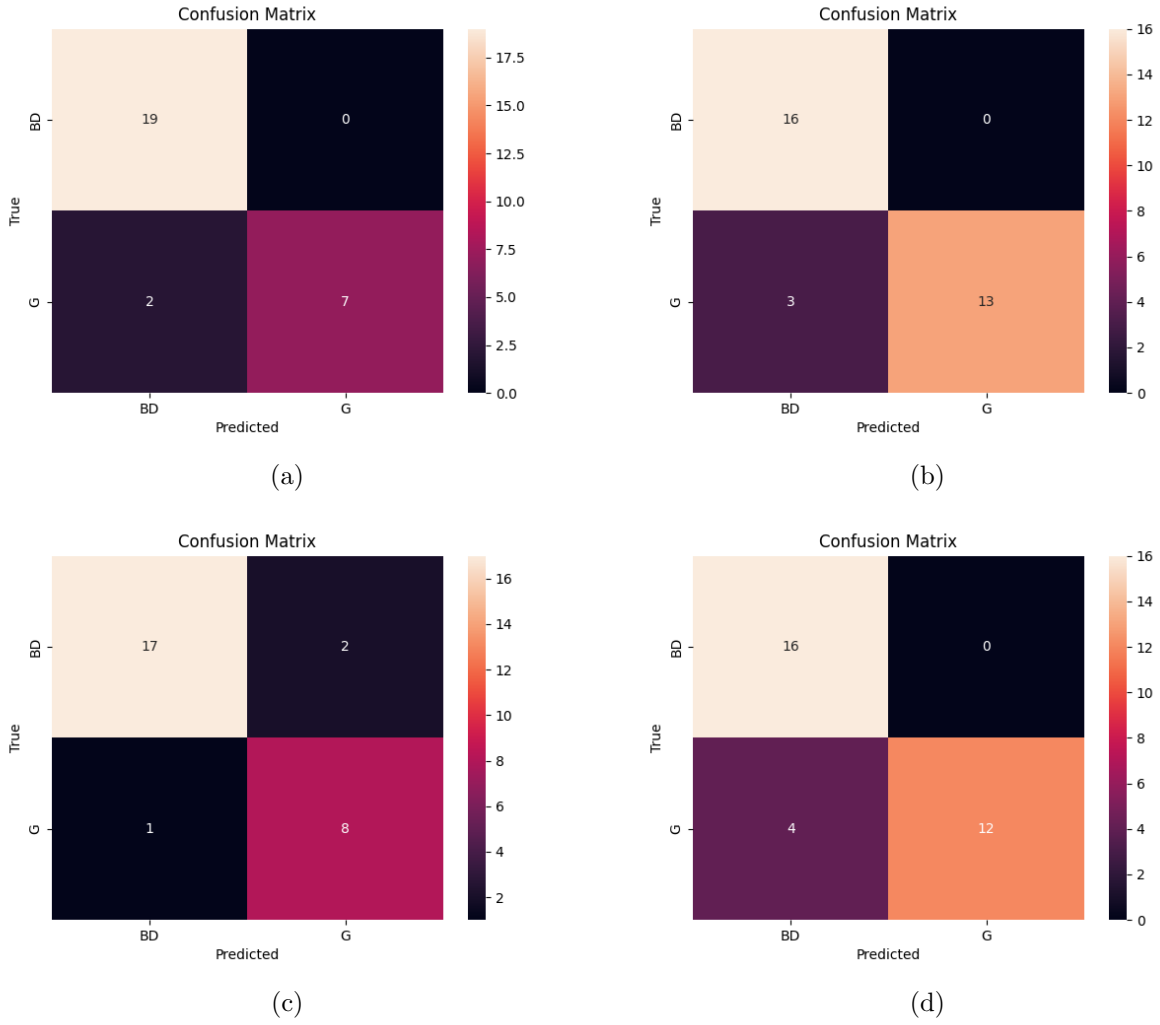


Figura 4.5: Matrici di confusione per il validation set ((a), (c)) e il test set ((b), (d)), ottenute con il modello Random Forest ((a), (b)) e Random Forest con SMOTE ((c), (d)). Per le magnitudini non rilevare è stato utilizzato l'approccio LLM.

sono stati rimossi dal dataset SHELLQs per garantire coerenza tra i set di training, validation e test. Anche in questo caso i parametri sono stati selezionati attraverso il `RandomizedSearchCV` (Tabella 4.1).

L'analisi comparativa tra i modelli testati evidenzia l'impatto delle diverse strategie di gestione delle magnitudini non rilevate (LLM vs. CBE) e dell'uso di SMOTE sul bilanciamento delle classi (Tabella 4.4).

L'approccio LLM si dimostra generalmente più efficace rispetto a CBE, soprattutto nel validation set. Con Random Forest senza SMOTE, la classe BD viene classificata con un F1-score del 95% nel validation set e del 91% nel test set, mentre la classe G ottiene rispettivamente 88% e 90%. Le metriche ponderate indicano un buon equilibrio complessivo, con un F1-score medio del 93% nel validation set e del 91% nel test set. La confusion matrix in Figura 4.5 mostra

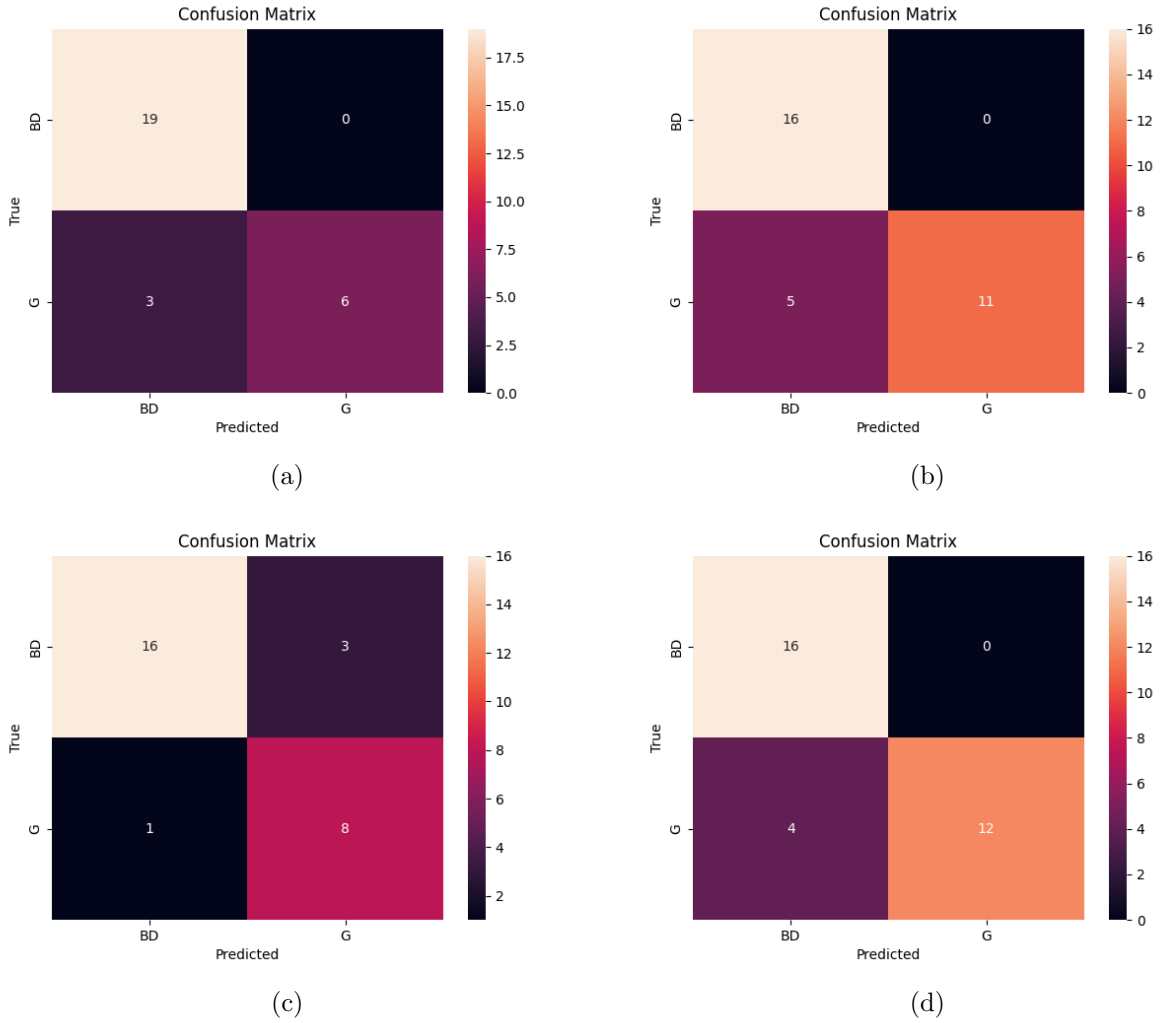


Figura 4.6: Matrici di confusione per il validation set ((a), (c)) e il test set ((b), (d)), ottenute con il modello Random Forest ((a), (b)) e Random Forest con SMOTE ((c), (d)). Per le magnitudini non rilevare è stato utilizzato l'approccio CBE.

come la maggior parte degli errori di classificazione avvenga nella classe G, con BD che viene generalmente classificata in modo più affidabile.

Quando si utilizza l'approccio CBE, invece, le performance globali sono leggermente inferiori. Con Random Forest senza SMOTE, la classe BD ottiene un F1-score del 93% nel validation set e dell'86% nel test set, mentre la classe G scende rispettivamente a 80% e 81%. L'F1-score medio è 89% nel validation set e 84% nel test set, suggerendo che l'approccio basato sui colori introduce maggiore variabilità nei dati, rendendo più difficile per il modello distinguere tra le classi, soprattutto nel test set. Come mostrato in Figura 4.6, gli errori si concentrano sulla classe G, con una maggiore confusione rispetto al caso LLM.
Questo potrebbe essere dovuto al fatto che le magnitudini non rilevate, trattate con LLM, risultano più simili alle magnitudini non censurate, rendendo il dataset più omogeneo agli occhi del modello. Al contrario, quando utilizziamo l'approccio CBE, questi valori possono

allontanarsi leggermente dalle altre magnitudini nel dataset, introducendo una variabilità che il modello potrebbe non interpretare in modo ottimale. Tuttavia, è importante sottolineare che, pur offrendo migliori performance in termini di classificazione, il metodo LLM non rispecchia il valore fisico reale delle magnitudini di questi oggetti, mentre la stima del metodo CBE è più aderente alla realtà astrofisica.

L'uso di SMOTE non porta a un miglioramento uniforme delle performance, ma introduce un trade-off tra precision e recall per la classe G.

Nell'approccio LLM, l'uso di SMOTE riduce leggermente l'accuratezza generale: l'F1-score medio scende dal 93% all'89% nel validation set e dal 91% all'87% nel test set. Per la classe BD, il recall rimane molto alto (100% nel test set), ma la precisione si riduce nel validation set (dal 90% al 94%), mentre per la classe G si osserva un calo di precisione sia nel validation set (da 100% a 80%) sia nel test set (da 81% a 75%). La Figura 4.5 evidenzia un aumento degli errori nella classe G, che viene spesso confusa con BD.

Nell'approccio CBE, l'effetto di SMOTE è simile: l'F1-score medio aumenta lievemente nel test set, passando da 84% a 87%, ma nel validation set si riduce da 89% a 86%. Il recall della classe BD rimane elevato, ma la precisione della classe G cala significativamente, passando da 100% a 73% nel validation set. La Figura 4.6 mostra come l'oversampling renda la classe BD più distinguibile, ma aumenti gli errori nella classe G.

Nel test set, le performance calano leggermente per entrambe le strategie, come atteso. Le confusion matrix riportate in Figura 4.5 e Figura 4.6 evidenziano che, nonostante il calo delle prestazioni, l'approccio LLM continua a mostrare una distribuzione più bilanciata degli errori, mentre la strategia con l'approccio CBE presenta un numero maggiore di falsi negativi per le galassie. Questo è dovuto al fatto che il dataset di test, J1030, è composto da oggetti più deboli, con un numero significativamente maggiore di limiti inferiori, soprattutto per le magnitudini i e Y, in particolare per le galassie. Questo aspetto è stato già discusso approfonditamente e riflette le difficoltà intrinseche nel classificare correttamente oggetti con molte informazioni censurate.

Nel complesso, il modello RF senza SMOTE con approccio LLM risulta il più stabile, con il miglior bilanciamento tra le classi. L'aggiunta di SMOTE migliora il recall della classe BD ma riduce la precisione della classe G, causando maggiore confusione. L'approccio CBE, invece, porta a performance leggermente inferiori rispetto a LLM, con una maggiore variabilità nei risultati e una classificazione meno precisa della classe G.

4.5 Random Forest con LDA (Linear Discriminant Analysis)

La combinazione di Random Forest (RF) e Linear Discriminant Analysis (LDA) è stata scelta per migliorare la separabilità delle classi e affrontare le discrepanze tra il training set (SHELLQs) e il test set (J1030). L'LDA riduce la dimensionalità dei dati trasformando le caratteristiche originali in componenti discriminanti che massimizzano la separazione tra le classi, favorendo una migliore generalizzazione del modello. Questo si è rivelato particolarmente utile come tecnica di domain adaptation, dato che il test set J1030 presenta una distribuzione significativamente diversa rispetto al training set SHELLQs.

L'implementazione dell'approccio combinato RF-LDA si è articolata in due fasi principali:

- **Trasformazione delle caratteristiche con LDA**: L'LDA è stata applicata come fase preliminare per ridurre le dimensioni del dataset e migliorare la separabilità delle classi. Utilizzando le caratteristiche originali (i, Y, z) e i colori derivati ($i - z$, $i - Y$, $z - Y$), i dati sono stati trasformati in componenti discriminanti, in modo da massimizzare la distanza tra le medie delle classi e ridurre la varianza intra-classe. Questo processo ha generato nuove caratteristiche più adatte per il successivo addestramento del modello (vedi capitolo 3.3).
- **Addestramento del Random Forest sui dati trasformati**: Dopo la trasformazione, il modello Random Forest è stato applicato seguendo la procedura descritta in precedenza, che comprende la ricerca e l'ottimizzazione dei parametri tramite `RandomizedSearchCV`. Questa procedura ha permesso di individuare configurazioni specifiche per ciascuna metodologia Monte Carlo utilizzata (Tabella 4.1).

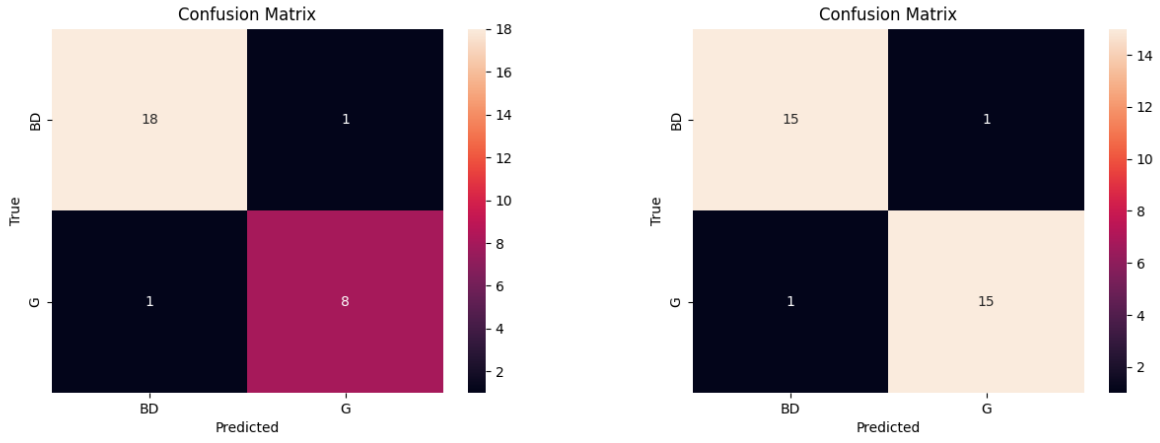
Il modello RF+LDA è stato addestrato sia sul dataset originale senza alcuna tecnica di oversampling, sia applicando SMOTE per bilanciare le classi. Queste combinazioni hanno permesso di valutare l'impatto della gestione delle magnitudini non rilevate e dell'oversampling sulle prestazioni del modello.

4.5.1 Performance di Random Forest con LDA con e senza resampling

I risultati riportati in Tabella 4.5 mostrano come la scelta tra l'approccio LLM e quello CBE influenzino le prestazioni del modello, sebbene in modo meno marcato rispetto ai risultati precedenti. In particolare, quando si utilizza l'approccio LLM, il modello RF + LDA ottiene un F1-score del 93% sul validation set e del 94% sul test set, con un bilanciamento quasi perfetto

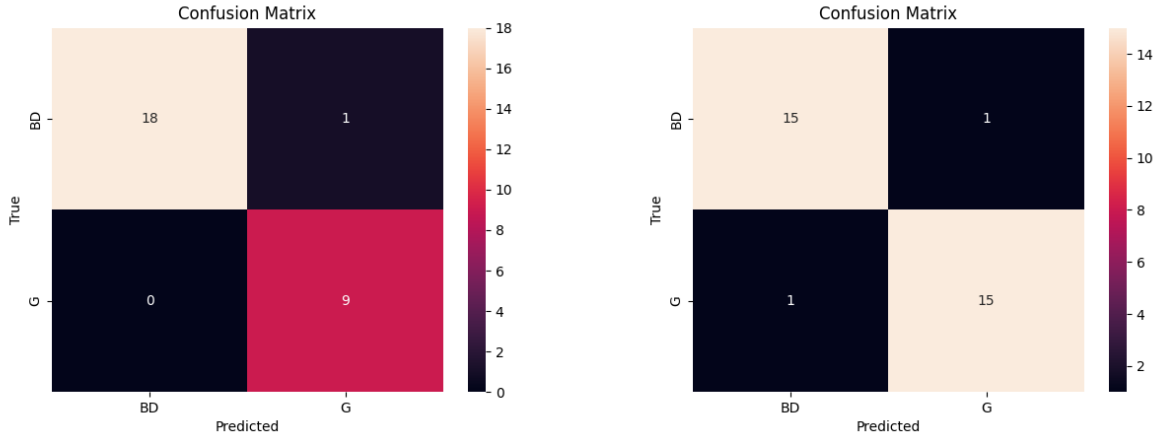
	Validation Set			Test Set		
	Precision	Recall	F1-score	Precision	Recall	F1-score
LLM						
RF + LDA						
BD	95%	95%	95%	94%	94%	94%
G	89%	89%	89%	94%	94%	94%
Weighted Avg	93%	93%	93%	94%	94%	94%
RF + SMOTE + LDA						
BD	100%	95%	97%	94%	94%	94%
G	90%	100%	95%	94%	94%	94%
Weighted Avg	97%	96%	96%	94%	94%	94%
CBE						
RF + LDA						
BD	95%	95%	95%	94%	94%	94%
G	89%	89%	89%	94%	94%	94%
Weighted Avg	93%	93%	93%	94%	94%	94%
RF + SMOTE + LDA						
BD	100%	84%	91%	88%	88%	88%
G	75%	100%	86%	88%	88%	88%
Weighted Avg	92%	89%	90%	88%	88%	88%

Tabella 4.5: Performance del modello Random Forest (RF) con LDA e SMOTE, confrontando le due diverse strategie per la gestione delle magnitudini non rilevate (LLM e CBE).



(a)

(b)



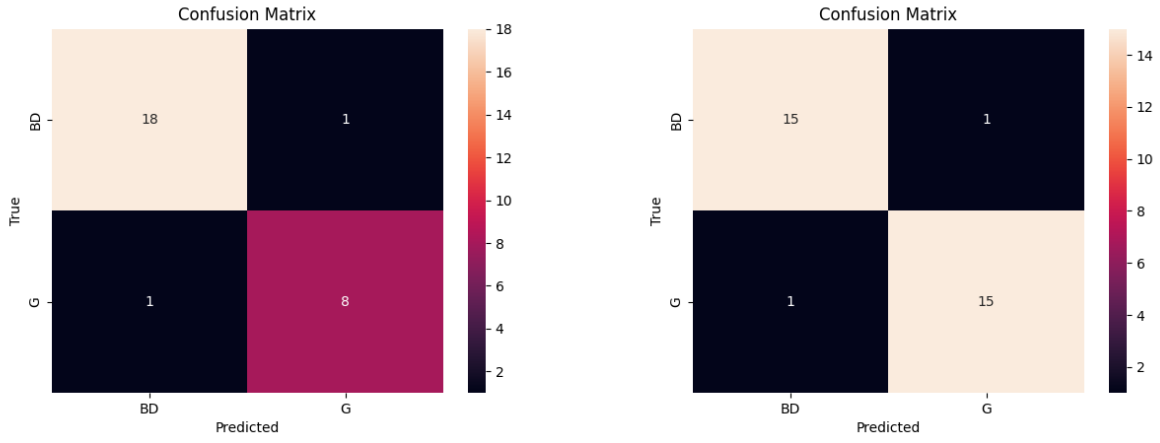
(c)

(d)

Figura 4.7: Matrici di confusione per il validation set ((a), (c)) e il test set ((b), (d)), ottenute con il modello Random Forest con LDA ((a), (b)) e Random Forest con LDA + SMOTE ((c), (d)). Per le magnitudini non rilevare è stato utilizzato l'approccio LLM.

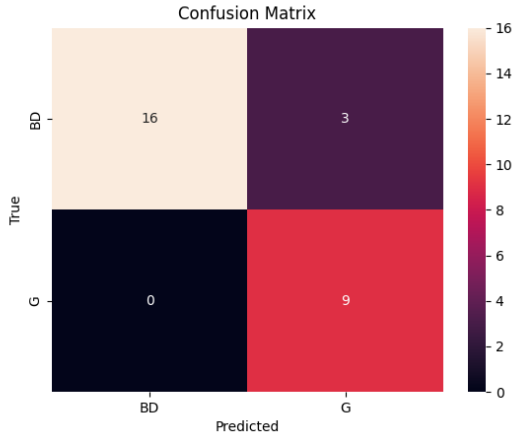
tra le classi Brown Dwarf (BD) e Galassie (G) (Figura 4.7). Anche in termini di accuratezza, questo approccio mostra ottime prestazioni, con un'accuracy del 94% sul validation set e del 93% sul test set.

L'inclusione di SMOTE in questo scenario porta a un leggero miglioramento dell'F1-score nel validation set (96%), con un recall del 100% per la classe G. Tuttavia, nel test set non si osservano miglioramenti significativi, poiché tutte le metriche rimangono al 94%. Questo suggerisce che, con l'LLM, l'oversampling può migliorare l'apprendimento delle caratteristiche nel training set senza compromettere le prestazioni in test, ma senza apportare un reale valore aggiunto. Anche l'accuratezza segue la stessa tendenza, con un incremento nel validation set (96%) e una stabilità nel test set (94%). Le confusion matrix riportate in **Figura 4.7** confermano che l'applicazione di SMOTE porta a un miglioramento nel bilanciamento delle classi, con un

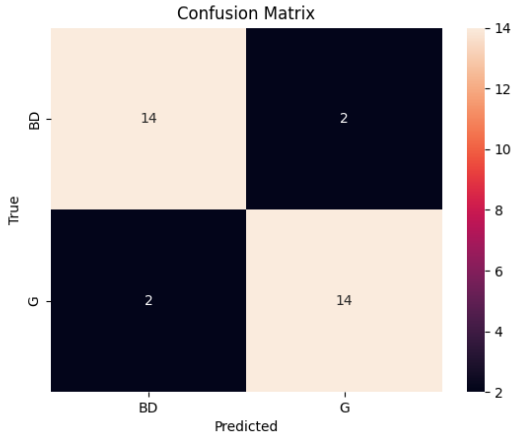


(a)

(b)



(c)



(d)

Figura 4.8: Matrici di confusione per il validation set ((a), (c)) e il test set ((b), (d)), ottenute con il modello Random Forest con LDA ((a), (b)) e Random Forest con LDA + SMOTE ((c), (d)). Per le magnitudini non rilevare è stato utilizzato l'approccio CBE.

recall più alto per la classe G nel validation set, ma senza variazioni nel test set.

Quando invece si utilizza l'approccio CBE le prestazioni di RF + LDA rimangono identiche a quelle ottenute con LLM (F1-score del 93% sul validation set e 94% sul test set). Come mostrato nelle confusion matrix in Figura 4.8, la distribuzione degli errori di classificazione non subisce variazioni significative rispetto al caso con LLM, indicando che entrambe le strategie portano a una separazione simile tra le classi. Anche l'accuratezza conferma questo trend, con valori rispettivamente del 93% e 94% per validation e test set. Questo risultato indica che, almeno in questo caso specifico, l'approccio CBE non introduce un beneficio chiaro rispetto all'uso di LLM.

L'aggiunta di SMOTE in questo contesto, invece, mostra un impatto negativo: nel validation set, la recall della classe BD scende al 84% e quello della classe G al 75%, con un conseguente

calo dell'F1-score complessivo al 90%. Anche nel test set le metriche si abbassano uniformemente all'88%, suggerendo che l'oversampling, in combinazione con l'approccio CBE, può introdurre rumore nei dati piuttosto che migliorare la capacità del modello di generalizzare. Coerentemente, anche l'accuratezza subisce un leggero calo, passando all'89% nel validation set e all'88% nel test set. Le confusion matrix in Figura 4.8 evidenziano come l'applicazione di SMOTE in questo scenario aumenti il numero di falsi positivi e falsi negativi, portando a una minore stabilità nelle prestazioni complessive.

In sintesi, i risultati mostrano che, a differenza di quanto osservato in precedenza, la scelta tra l'approccio LLM e quello CBE non influisce in modo significativo sulle prestazioni del modello RF + LDA. Entrambe le strategie portano a risultati simili, con un F1-score stabile tra validation e test set. Tuttavia, l'applicazione di SMOTE mostra effetti contrastanti: migliora leggermente la recall della classe G quando si usa l'approccio LLM, ma peggiora le prestazioni quando si adottano il CBE. Questo suggerisce che, in un dataset relativamente bilanciato, l'oversampling non sempre è necessario e può persino risultare controproducente. I valori di accuratezza riportati in Tabella 4.3 confermano questa osservazione, evidenziando come il modello RF + LDA con LLM mantenga un'accuratezza elevata sia nel validation (94%) che nel test set (93%), mentre l'uso di SMOTE porta a un lieve miglioramento nel validation set (96%) ma senza variazioni nel test (94%). Viceversa, con l'approccio CBE, l'accuratezza rimane stabile senza SMOTE (93%-94%), ma subisce un calo con l'oversampling, scendendo all'89% nel validation set e all'88% nel test set.

4.6 Probabilistic Random Forest

L'implementazione del modello Probabilistic Random Forest (PRF) è stata adottata come soluzione per tenere conto delle incertezze nei dati, inclusi gli errori associati alle magnitudini e ai colori. A differenza del Random Forest tradizionale, che tratta i dati come deterministici, il PRF è progettato per gestire in modo esplicito le incertezze e le variazioni nei dati osservativi. Questo ha reso il PRF l'algoritmo ideale per includere le incertezze nei dati, evitando la necessità di applicare il metodo Monte Carlo.

Invece di applicare Monte Carlo, il processo ha incluso la stima delle magnitudini non rilevate (attraverso LLM e CBE) e il calcolo degli errori sui colori tramite la propagazione degli errori. Questa modalità ha consentito al PRF di incorporare le incertezze in modo nativo durante il processo di classificazione, migliorando la robustezza del modello e la sua capacità predittiva.

Poiché il PRF non è compatibile con tecniche standard di ricerca automatica dei parametri, come il `GridSearchCV`, la selezione dei parametri è stata effettuata utilizzando una funzione

dedicata, `random_grid_prf` (vedere Appendice C), progettata per esplorare casualmente una griglia predefinita di iperparametri.

La funzione ha generato combinazioni casuali di parametri seguendo il seguente schema:

- **Numero di alberi (`n_estimators`)**: scelti tra 10 e 100 con incrementi di 10.
- **Profondità massima degli alberi (`max_depth`)**: scelta tra 3 e 15.

La funzione `random_grid_prf` genera un set casuale di iperparametri per ciascun modello, che viene addestrato sul training set e valutato sulla base delle sue prestazioni.
Questa procedura ha consentito di adattare il modello PRF ai dati disponibili, sfruttando l'incorporazione delle incertezze nei colori e nelle magnitudini attraverso l'architettura nativa del modello.

Dopo l'esplorazione casuale degli iperparametri, i parametri finali selezionati per l'addestramento del modello PRF sono riportati nella Tabella 4.1.

Questi parametri sono stati utilizzati per addestrare il modello finale sul dataset SHELLQs, mentre le sue prestazioni sono state valutate sul test set J1030. In particolare, il PRF è stato addestrato sia su un dataset non bilanciato sia su una versione bilanciata con SMOTE, permettendo di analizzare l'impatto dell'oversampling sintetico sulle performance del modello. Inoltre, come già accennato per le magnitudini non rilevate, sono stati considerati i due approcci distinti LLM e CBE descritti precedentemente generando un unico valore per ogni magnitudine non rilevata (a differenza dell'approccio Monte Carlo, che prevede la generazione di più campioni). Questa analisi ha permesso di valutare come diverse strategie di gestione delle magnitudini non rilevate influenzino le prestazioni del modello, fornendo un quadro più completo delle possibili fonti di incertezza e del loro impatto sulla classificazione.

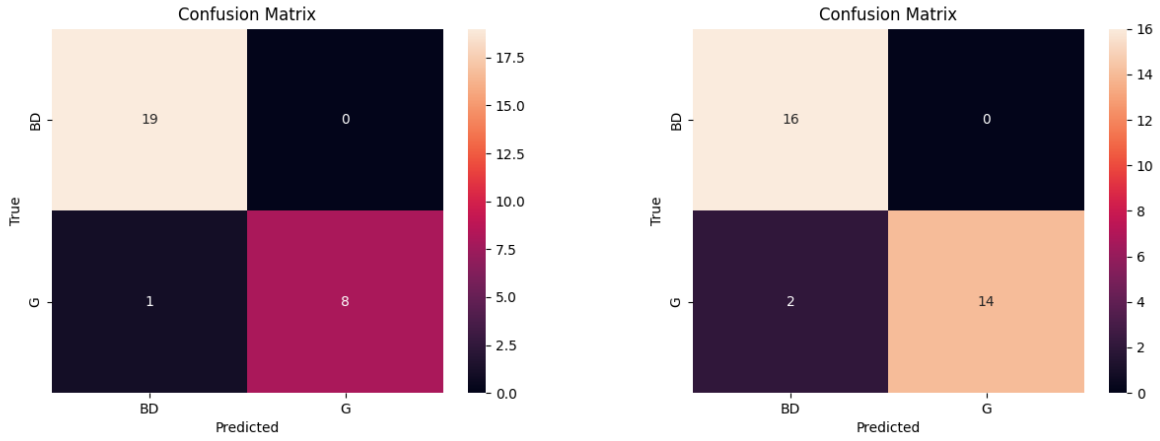
Il PRF si è rivelato particolarmente utile per affrontare dataset caratterizzati da incertezze nei dati osservativi, come nel caso del training set (SHELLQs) e del test set (J1030). L'assenza di Monte Carlo ha semplificato il flusso di preprocessing. Il modello ha dimostrato una robustezza intrinseca nell'affrontare la complessità del problema, migliorando la capacità predittiva senza richiedere adattamenti aggiuntivi per incorporare gli errori.

4.6.1 Performance di PRF con e senza resampling

Dai risultati in Tabella 4.6 possiamo notare alcune differenze interessanti tra le varie strategie adottate per gestire le magnitudini non rilevate. Quando utilizziamo l'approccio LLM, il modello mostra ottime prestazioni, con un F1-score del 96% nel validation set e del 94% nel test set,

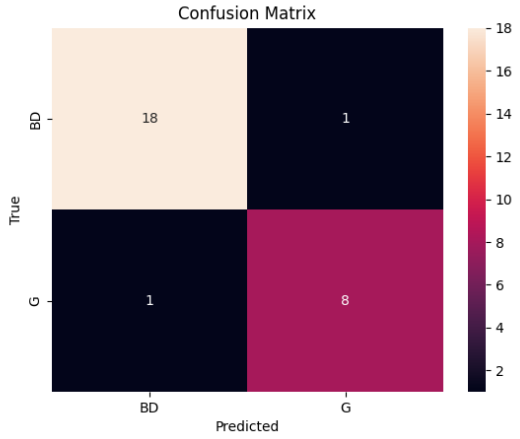
	Validation Set			Test Set		
	Precision	Recall	F1-score	Precision	Recall	F1-score
LLM						
PRF						
BD	95%	100%	97%	89%	100%	94%
G	100%	89%	94%	100%	88%	93%
Weighted Avg	97%	96%	96%	94%	94%	94%
PRF + SMOTE						
BD	95%	95%	95%	84%	100%	91%
G	89%	89%	89%	100%	81%	90%
Weighted Avg	93%	93%	93%	92%	91%	91%
CBE						
PRF						
BD	90%	95%	92%	80%	100%	89%
G	88%	78%	82%	100%	75%	86%
Weighted Avg	89%	89%	89%	90%	88%	87%
PRF + SMOTE						
BD	95%	100%	97%	94%	100%	97%
G	100%	89%	94%	100%	94%	97%
Weighted Avg	97%	96%	96%	97%	97%	97%

Tabella 4.6: Performance del modello PRF e PRF+SMOTE, confrontando le due diverse strategie per la gestione delle magnitudini non rilevate (LLM e CBE).

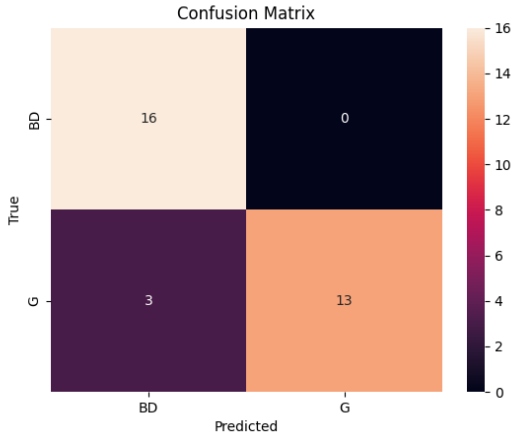


(a)

(b)



(c)

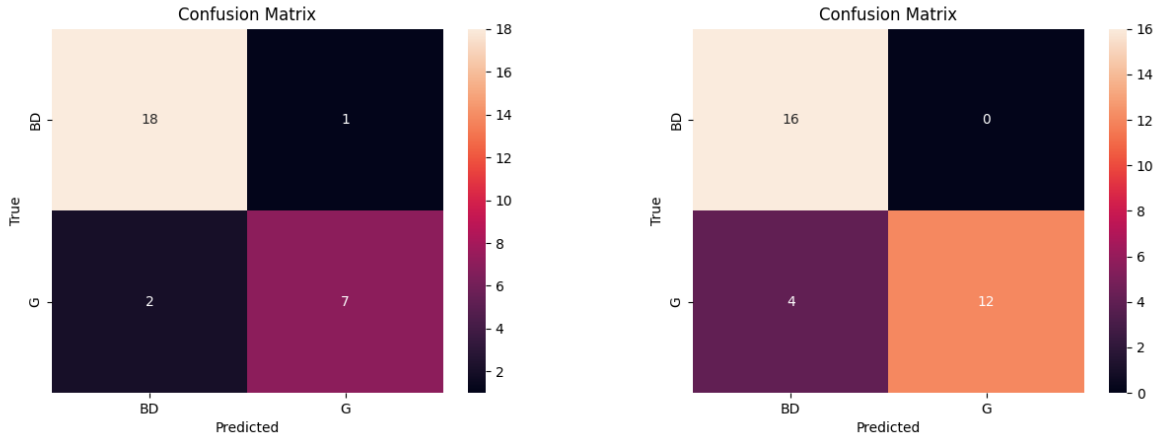


(d)

Figura 4.9: Matrici di confusione per il validation set ((a), (c)) e il test set ((b), (d)), ottenute con il modello PRF ((a), (b)) e PRF + SMOTE ((c), (d)). Per le magnitudini non rilevare è stato utilizzato l'approccio LLM.

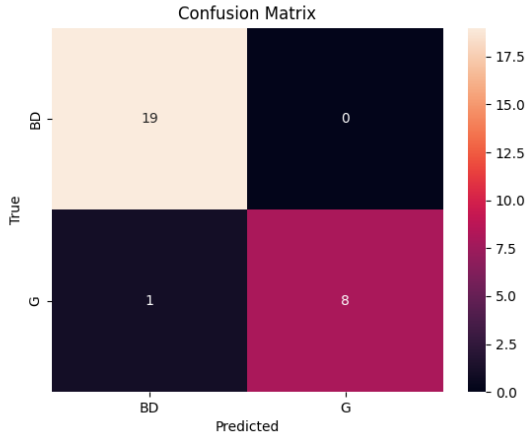
supportato da un'accuracy rispettivamente del 96% e del 94% (Tabella 4.3). Questo significa che il modello riesce a distinguere bene tra Brown Dwarf (BD) e Galassie (G), mantenendo un buon equilibrio tra precision e recall. La recall dei BD è particolarmente alto (100% in entrambi i set), indicando che tutti gli oggetti di questa classe vengono riconosciuti correttamente. Le Galassie, invece, hanno un recall leggermente più basso (89% nel validation set e 88% nel test set), il che suggerisce che qualche galassia venga confusa con un BD. Nel complesso, questo approccio sembra essere molto efficace. Tuttavia, va considerato che l'uso dell'LLM potrebbe non rispecchiare accuratamente i valori fisici di queste sorgenti, a differenza dell'approccio CBE, che invece forniscono una stima più aderente alla realtà fisica degli oggetti. Le confusion matrix in Figura 4.9 mostrano come il modello classifica le sorgenti con questo approccio, evidenziando gli errori di classificazione tra le due classi.

L'aggiunta di SMOTE non porta a un miglioramento netto rispetto alla versione senza

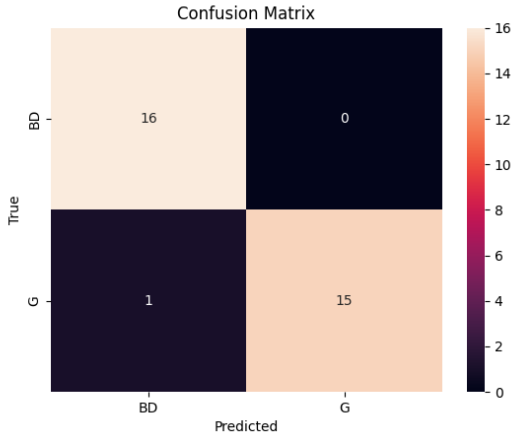


(a)

(b)



(c)



(d)

Figura 4.10: Matrici di confusione per il validation set ((a), (c)) e il test set ((b), (d)), ottenute con il modello PRF ((a), (b)) e PRF + SMOTE ((c), (d)). Per le magnitudini non rilevate è stato utilizzato l'approccio CBE.

oversampling. L'F1-score medio scende leggermente nel test set (91% contro il 94% precedente), con un'accuracy che passa dal 94% al 91%, e la recall delle Galassie si riduce all'81%, il che significa che il modello ha più difficoltà nel riconoscerle. Tuttavia, la recall dei BD rimane 100%, quindi tutti gli oggetti di questa classe vengono ancora identificati correttamente. SMOTE sembra aver portato a un bilanciamento delle classi nel training set, ma nel test set il modello non riesce a mantenere le stesse performance della versione senza SMOTE. Questo potrebbe essere dovuto al fatto che il test set, J1030, presenta un numero maggiore di limiti inferiori rispetto al validation set (SHELLQs), essendo composto da sorgenti più deboli.

Quando i valori censurati vengono stimati utilizzando l'approccio CBE, le prestazioni calano leggermente. Nel validation set, l'F1-score medio scende all'89%, e nel test set si abbassa ulteriormente all'87%. L'accuracy passa rispettivamente all'89% e all'88%, confermando un leggero calo rispetto all'approccio LLM. La recall delle Galassie in particolare è più basso (78%

nel validation set e solo 75% nel test set), il che significa che il modello fatica maggiormente a distinguere questa classe. Questo risultato suggerisce che l'approccio CBE introduce un po' di incertezza. Tuttavia, rispetto all'LLM, questa metodologia riflette meglio la distribuzione dei parametri fisici delle sorgenti censurate. Le confusion matrix in Figura 5.3 evidenziano come questa strategia porti a un maggior numero di errori di classificazione, in particolare per le Galassie.

L'aggiunta di SMOTE in questo caso sembra invece avere un effetto positivo: nel test set, l'F1-score medio raggiunge il 97%, con una netta crescita rispetto alla versione senza SMOTE, e un'accuracy che aumenta fino al 97% (Tabella 4.3). In particolare, la recall delle Galassie aumenta fino al 94%, il che significa che il modello riesce a identificarle molto meglio rispetto alla versione senza oversampling. Questo suggerisce che, quando si usa l'approccio CBE, l'aumento dei dati sintetici aiuta il modello a generalizzare meglio.

I risultati mostrano che il trattamento dei dati censurati ha un impatto significativo sulle prestazioni del modello. Il metodo LLM sembra essere la scelta più affidabile in termini di performance, soprattutto senza SMOTE, ma non riflette accuratamente la distribuzione fisica delle sorgenti. Se si sceglie di stimare le magnitudini non rilevate usando i colori, allora SMOTE aiuta a compensare la variabilità introdotta dalla stima, migliorando le prestazioni complessive e rendendo il modello più aderente alla realtà astrofisica.

4.7 Confronto tra i modelli

Dai risultati ottenuti, emerge un quadro chiaro sull'impatto delle diverse strategie adottate per migliorare la classificazione delle sorgenti nei dataset SHELLQs e J1030. Il confronto tra i vari modelli ci permette di evidenziare quali scelte abbiano avuto un impatto positivo sulle prestazioni e quali, invece, non abbiano portato ai miglioramenti sperati.

Un primo aspetto fondamentale riguarda il trattamento delle magnitudini censurate. Quando applichiamo l'LLM, il modello Random Forest, senza particolari modifiche, ottiene risultati molto solidi, con un F1-score attorno al 93% nel validation set e al 91% nel test set, accompagnato da una accuracy del 93% e del 91% rispettivamente. Applicando invece il CBE, le prestazioni risultano leggermente inferiori, con un calo più evidente nel test set, dove l'accuracy scende all'84%. Questo suggerisce che, pur offrendo una rappresentazione più fisicamente accurata delle sorgenti, questa strategia introduce maggiore incertezza nel modello, rendendo più difficile la separazione tra le classi.

Un altro elemento interessante è il ruolo del bilanciamento delle classi tramite SMOTE. In teoria, questa tecnica dovrebbe aiutare il modello a migliorare il riconoscimento delle classi

meno rappresentate, ma i risultati mostrano un quadro più complesso. Quando si utilizza l'LLM, l'aggiunta di SMOTE non porta reali benefici: anzi, nel test set l'accuracy scende dal 91% all'89%, segno che l'oversampling potrebbe aver introdotto più rumore che vantaggi. La situazione cambia leggermente quando si utilizza l'approccio CBE: in questo caso, SMOTE sembra avere un effetto positivo, migliorando le prestazioni e aumentando la capacità del modello di distinguere meglio le Galassie nel test set, con un aumento dell'accuracy dal 84% all'88%.

L'uso di LDA si rivela invece una strategia molto efficace. Applicando questa tecnica, le prestazioni del modello diventano molto più stabili, con un F1-score che rimane costantemente alto sia nel validation set che nel test set, intorno al 93-94%, mentre l'accuracy si attesta sul 94% per entrambi i set di dati. Questo suggerisce che la riduzione dimensionale operata dalla LDA aiuta la separazione tra le classi e contribuisce a mitigare le incertezze nei dati, migliorando la capacità del modello di generalizzare su nuovi campioni.

Il miglior risultato si ottiene con il PRF, il quale, già senza alcun bilanciamento, supera le performance di Random Forest, con un F1-score che raggiunge il 96% nel validation set e il 94% nel test set, supportato da un'accuracy di pari valore (96% e 94%). Questo dimostra che PRF è in grado di gestire meglio le incertezze dei dati rispetto a un semplice Random Forest. Quando poi si combina PRF con SMOTE, le prestazioni migliorano ulteriormente, raggiungendo il punteggio più alto in assoluto nel test set, con un F1-score medio del 97% e un'accuracy del 97%. Questo risultato suggerisce che, quando si stimano le magnitudini attraverso distribuzioni troncate, l'oversampling può effettivamente compensare le incertezze introdotte dalla stima e migliorare la generalizzazione del modello.

In definitiva, i risultati mostrano che non esiste una strategia perfetta, ma che la scelta dipende molto dal modo in cui vengono trattati i dati censurati. Se si preferisce un approccio più conservativo e meno soggetto a ipotesi sulla distribuzione delle magnitudini, il modello Random Forest con LDA senza SMOTE offre prestazioni stabili e affidabili, con un'accuracy del 94%. Se invece si vuole sfruttare al massimo l'informazione disponibile, stimando le magnitudini censurate e bilanciando il dataset con SMOTE, allora PRF si conferma la scelta migliore, raggiungendo le prestazioni più elevate con un'accuracy del 97% nel test set.

Capitolo 5

Selezione e Classificazione delle Galassie nel Campo di J0050

In questo capitolo presentiamo il processo di selezione e classificazione dei candidati galattici identificati nel campo J005006.67+344521.6 (J0050). Il quasar J0050+3445 è stato scoperto nell'ambito del Canada-France High-z Quasar Survey (CFHQS; Willott et al. (2010b)) ed è uno dei quasar più luminosi del CFHQS, con magnitudine $z = 20.5$. È anche uno degli oggetti più distanti del campione CFHQS, con un redshift $z = 6.253 \pm 0.003$ misurato dalla riga di emissione Mg II (Willott et al. (2010a)). La riga di emissione Mg II è stata utilizzata anche per una stima viriale della massa del buco nero, $M_{BH} = 2.610^9 M\odot$. Questa fase rappresenta l'applicazione finale del lavoro svolto, in cui vengono applicati i criteri di selezione definiti nei capitoli precedenti e analizzati i risultati ottenuti tramite diversi algoritmi di machine learning. L'obiettivo è valutare l'efficacia dei modelli sviluppati nell'identificare galassie Lyman-break ad alto redshift e comprendere le potenzialità e i limiti delle metodologie adottate.

Anzitutto analizzeremo il processo di selezione degli oggetti su cui verranno effettuate le predizioni, evidenziando le caratteristiche fotometriche utilizzate e i criteri applicati per individuare i candidati più promettenti. Successivamente, discuteremo i risultati delle classificazioni ottenute con i vari modelli, confrontando le performance e interpretando i casi più significativi. Infine, trarremo le conclusioni sull'efficacia delle strategie adottate, individuando possibili miglioramenti e sviluppi futuri per affinare ulteriormente l'identificazione di galassie ad alto redshift in campi profondi.

5.1 Selezione dei candidati

I dati attorno al quasar J0050 provengono da osservazioni con LBC di circa 1.5 ore con i filtri i e z, e immagini simultanee per 3 ore con il filtro r, al fine di raggiungere una magnitudine limite di $z = 25.2$ a 5σ, e $i = 26.6$ e $r = 27.2$ a $\sim 5\sigma$. Il rilevamento nelle ultime due bande è

Banda	z_p	$\sigma_{background}$[ADU/pixel]	$\overline{\Delta Mag} \pm dMag_{err}$	$\text{mag}_{2\sigma}$
z	27.2	0.16	0.11 ± 0.02	26.75
i	27.57	0.07	0.11 ± 0.06	27.47
r	27.75	0.03	0.15 ± 0.02	27.97
Y	24.59	0.06	0.20 ± 0.03	24.44

Tabella 5.1: Valori fotometrici per le diverse bande: z_p indica lo zero-point; $\sigma_{background}$ indica la deviazione standard del rumore di fondo; $\overline{\Delta Mag}$ rappresenta il valore medio della differenza tra MAG_APER e MAG_AUTO e viene usata per correggere il valore della magnitudine di apertura per gli effetti del seeing di ciascuna banda; infine, $mag_{2\sigma}$ corrisponde al valore limite della magnitudine a 2σ.

stato eseguito in modalità duale, utilizzando le posizioni delle galassie rivelate nella banda z come prior per aumentare la significatività della rilevazione nelle altre bande e per avere una misura del colore $i - z$ (ad esempio) per più oggetti possibili.

La riduzione, calibrazione fotometrica, validazione qualitativa e correzione astrometrica delle immagini sono state effettuate dal team LBT/LBC presso il "LBC survey center" a Roma[1]. Da queste immagini, per ogni banda, gli oggetti sono stati estratti utilizzando il software SExtractor[2], imponendo che un oggetto fosse considerato tale solo se il segnale in almeno 9 pixel consecutivi supera la soglia di 1.5σ di significatività rispetto al background, per una significatività totale della detection $\sqrt{N_{pix}}\sigma_{pix} = \sqrt{9} \times 1.5 = 4.5\sigma$. Questi valori sono gli stessi usati per altri campi SQUEEzE (Vito et al. in prep). Inoltre,il software ha ricevuto in input gli zero points[3] Z_p, selezionati dalle tabelle informative di LBT/LBC e riportati nella Tabella 5.1, per convertire il flusso rivelato dallo strumento misurato in magnitudini apparenti secondo la relazione:

$$MAG = -2.5 log_{10}(\text{count rate}) + Z_p \tag{5.1}$$

Il processo di rivelazione degli oggetti ha prodotto tabelle per ogni banda contenenti vari parametri, tra cui:

- NUMBER: identificativo univoco per ciascun oggetto;
- ALPHA_J2000 e DELTA_J2000: coordinate celesti (RA e DEC);
- MAG_APER e MAGERR_APER: magnitudine e incertezza associate a un'apertura circolare fissa, che abbiamo imposto essere 10 pixel di raggio. Questo valore è stato scelto

[1] `https://lsc.oa-roma.inaf.it/`

[2] Per ulteriori dettagli visitare il sito `https://sextractor.readthedocs.io/en/latest/Introduction.html`

[3] La magnitudine alla quale il flusso è pari a 1 fotone/s.

come trade-off tra includere più flusso possibile (considerando anche il seeing) ed evitare overlap con altre sorgenti.

- MAG_AUTO e MAGERR_AUTO: magnitudine e incertezza ottenute con un'apertura ellittica adattiva di Kron, determinata automaticamente in base alla forma e alla dimensione dell'oggetto, permettendo una stima più accurata del flusso totale, per sorgenti brillante e isolate.

- FLAGS: parametro che assume valori diversi da zero quando alcuni pixel nell'immagine sono considerati inaffidabili (ad esempio, in presenza di sorgenti vicine e non completamente separate).

Il seeing può allargare la distribuzione della luce di un oggetto, rendendo un'apertura fissa potenzialmente inadatta a catturare l'intero flusso. Questo può portare a una sottostima della magnitudine. Nel caso di sorgenti puntiformi, infatti, la luce segue una Point Spread Function (PSF), il che significa che una parte significativa del flusso può trovarsi nelle code della distribuzione e quindi al di fuori dell'apertura scelta, portando a una sottostima della magnitudine. L'apertura ellittica adattiva di Kron, utilizzata per il calcolo di MAG_AUTO, si adatta alla forma dell'oggetto, tenendo conto della sua PSF, permettendo di catturare una frazione maggiore del flusso totale. Idealmente, questa tecnica mira a includere l'intero flusso dell'oggetto, anche se nella pratica una parte residua può comunque andare persa. Tuttavia, rispetto alle aperture fisse, l'apertura di Kron è quella che si avvicina di più a una misura completa del flusso totale.

Per correggere gli effetti del seeing è stata quindi calcolata la differenza tra le due magnitudini, così da compensare la perdita di flusso dovuta all'uso di un'apertura circolare fissa:

$$\Delta\text{Mag} = \text{MAG\_APER} - \text{MAG\_AUTO} \tag{5.2}$$

e successivamente il valore medio $\overline{\Delta\text{Mag}}$, riportato in Tabella 5.1.

In fine, per ciascun oggetto in ogni banda, MAG_APER è stata corretta utilizzando $\overline{\Delta\text{Mag}}$, secondo la relazione:

$$\text{MAG\_APER\_CORR} = \text{MAG\_APER} - \overline{\Delta\text{Mag}} \tag{5.3}$$

Questo approccio riduce la sottostima della magnitudine che si verifica con aperture fisse, migliorando l'accuratezza della misura fotometrica.

Poiché l'obiettivo è identificare galassie ad alto redshift, sono stati selezionati gli oggetti rilevati in banda z e nelle altre bande. Il catalogo iniziale, utilizzando la banda z come prior, comprendeva 21.967 oggetti.
Per valutare la distribuzione delle magnitudini nei diversi filtri, abbiamo generato istogrammi

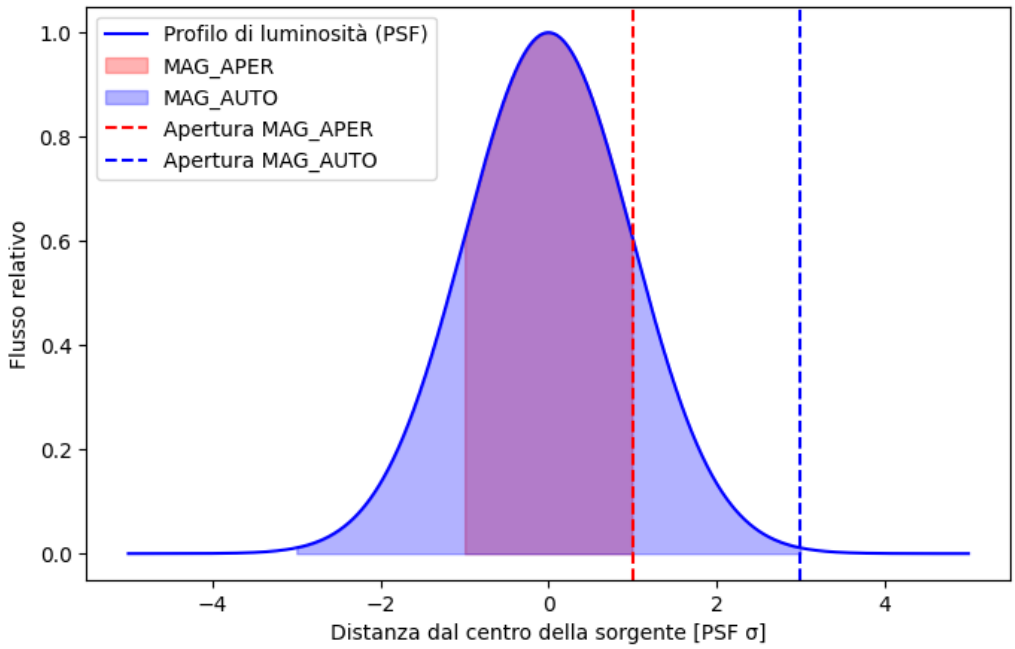


Figura 5.1: Differenza nel flusso (e quindi nella magnitudine) catturato dalla PSF utilizzando un'apertura circolare fissa e un'apertura ellittica adattiva di Kron.

nelle bande r, i, z e Y. Questi istogrammi permettono di visualizzare la distribuzione della luminosità degli oggetti nelle diverse bande e identificare eventuali caratteristiche distintive tra le popolazioni osservate.

Successivamente, è stata determinata la magnitudine limite a 2σ ($\mathrm{mag}_{2\sigma}$, riportata in Tabella 5.1). Per tutti gli oggetti con MAG_APER_CORR $> \mathrm{mag}_{2\sigma}$, è stata assegnata la magnitudine limite $\mathrm{mag}_{2\sigma}$.

È stata quindi costruita un'unica tabella contenente tutte le magnitudini corrette, le coordinate RA e DEC, e i colori $i-z$ e $z-Y$, con i relativi errori calcolati tramite propagazione.

A questo punto, sono stati applicati i seguenti criteri di selezione:

- $r > mag_{2\sigma}$ (oggetti senza significativa emissione in banda r);
- $i - z > 1.5$;
- $z - Y < 0.46(i - z) - 0.6$

Abbiamo adottato i tagli sui colori seguendo Pudoka et al. (2024), che utilizza lo stesso strumento impiegato in questo lavoro. In questo modo è possibile distinguere la maggior parte delle LBGs dai BDs, poiché questi ultimi appaiono più rossi rispetto alle galassie ad alto redshift nel colore $z - Y$ (si rimanda alla sezione 1.2). Dopo questa selezione, il numero di oggetti si è ridotto a 251. Infine, questi oggetti sono stati visualizzati sulle immagini in tutte le bande (Figure 5.4 e 5.4) per escludere artefatti. Il campione finale è composto da 7 oggetti (Tabella 5.2).

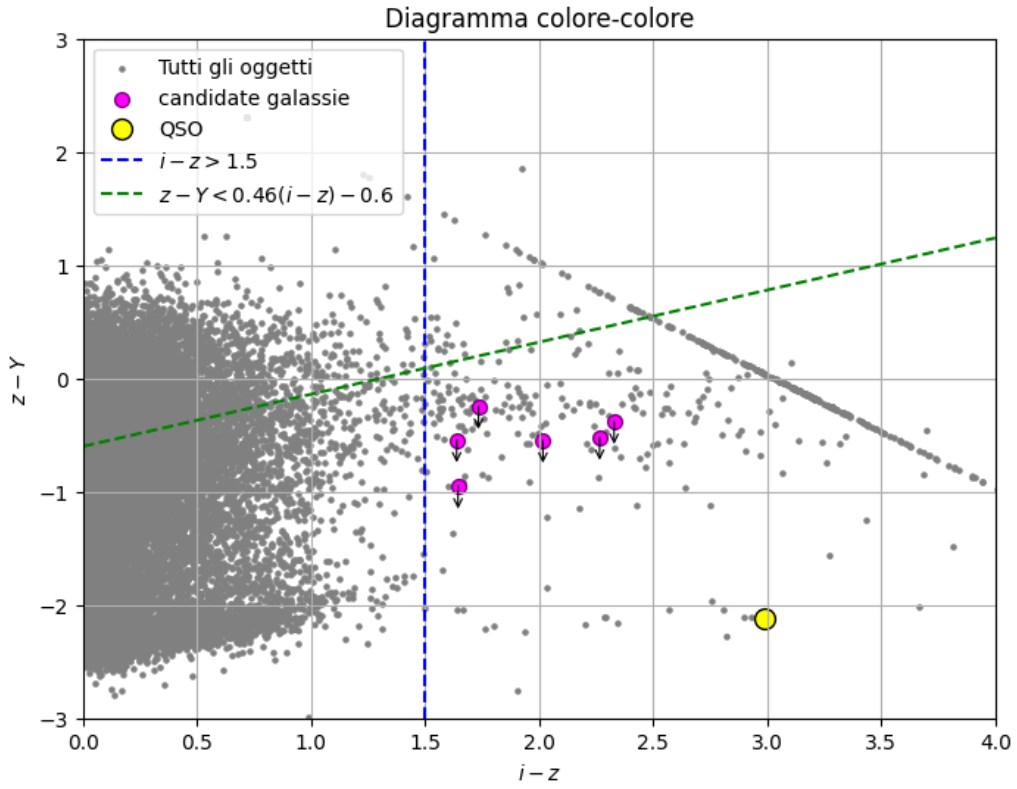


Figura 5.2: Diagramma colore-colore con le candidate LBG (punti magenta) e il quasar (punto giallo). Le frecce in corrispondenza dei punti magenta mostrano i limiti superiori $(z-Y)$ laddove la sorgente non è rilevata in banda Y. Questo mostra il colore $z-Y$ rispetto al colore $i-z$. La linea verticale in blu mostra il taglio in colore $(i-z>1.5)$, mentre la linea diagonale in verde mostra il taglio $z-Y<0.46(i-z)-0.6$. I punti grigi mostrano tutte le sorgenti rilevate nel campo J0050.

Queste sette sorgenti sono state utilizzate per effettuare predizioni tramite algoritmi di machine learning.

5.2 Risultati delle predizioni

Per effettuare le predizioni sul dataset composto dalle sette candidate selezionate, è stato incluso anche il colore $i-Y$, poiché tutti gli algoritmi addestrati prevedevano questa caratteristica tra le variabili di input.

Le predizioni sono state effettuate utilizzando diversi algoritmi, al fine di confrontare i risultati ottenuti. In particolare, sono stati impiegati i seguenti modelli: RF + LDA, RF + SMOTE e PRF + SMOTE, tutti addestrati utilizzando l'approccio CBE per le magnitudini non rilevate. Inoltre, le predizioni sono state effettuate sia utilizzando il colore medio delle galassie,

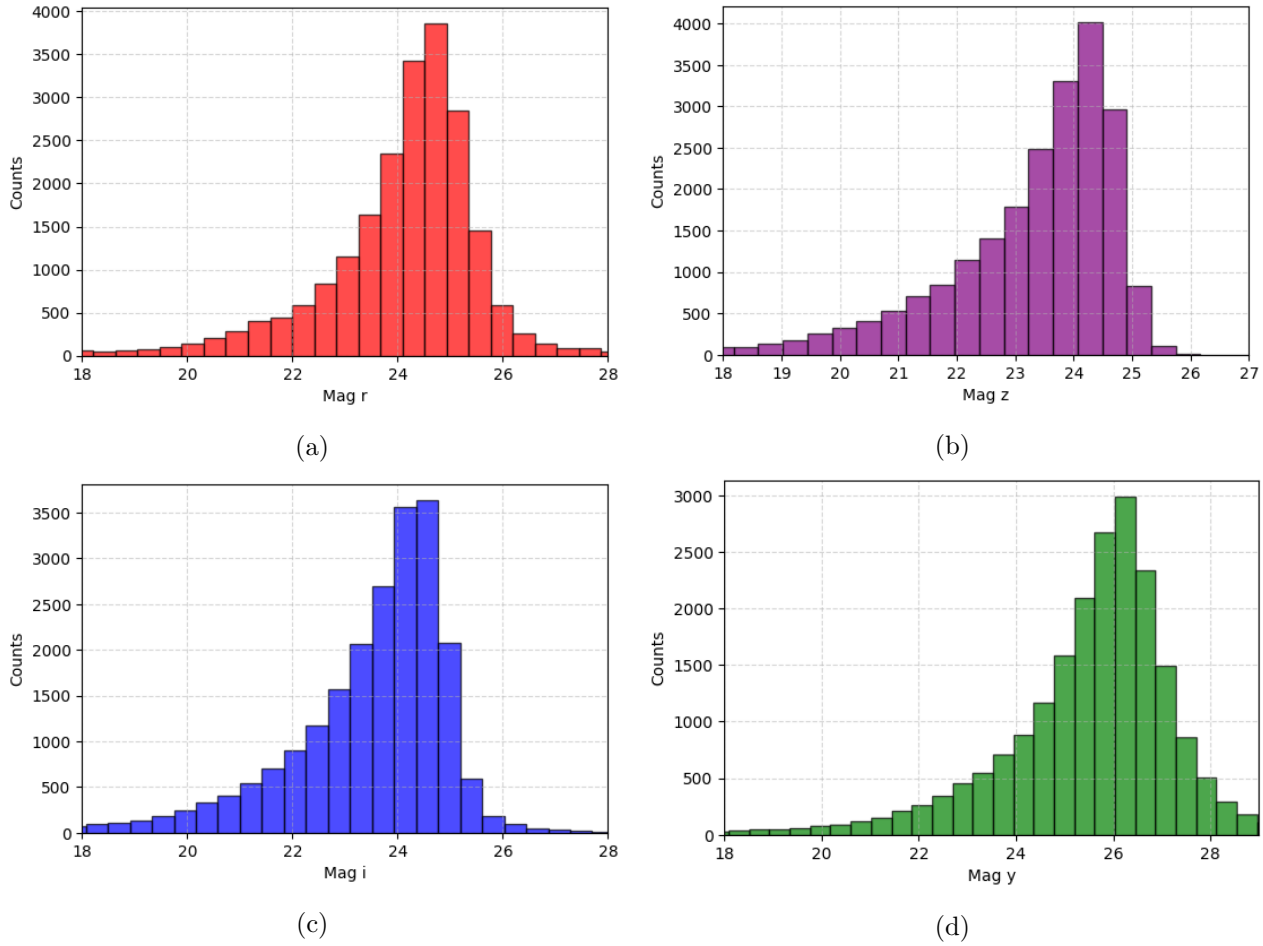


Figura 5.3: Istogrammi per le magnitudini r(a), z(b), i(c) e Y(d) per tutti gli oggetti del campo di J0050.

sia utilizzando quello delle brown dwarfs. La scelta di questi tre algoritmi si basa sulle loro elevate prestazioni e capacità di bilanciare le classi:

- RF + SMOTE: ha dimostrato buone performance, con un F1-score tra l'86% e l'89%, garantendo un miglior bilanciamento tra le classi rispetto alla Random Forest standard. L'uso di SMOTE ha contribuito a ridurre il rischio che il modello trascurasse la classe minoritaria, migliorando la capacità di generalizzazione.

- RF + LDA: ha ottenuto precisione, recall e F1-score compresi tra il 93% e il 94% per entrambe le classi. L'applicazione della Linear Discriminant Analysis (LDA) ha permesso di ridurre la dimensionalità massimizzando la separazione tra le classi, favorendo una migliore generalizzazione del modello e mantenendo un'accuratezza ben bilanciata.

- PRF + SMOTE: è il modello con le prestazioni globali più elevate (97% su tutte le metriche). L'uso del PRF (Probabilistic Random Forest) combinato con SMOTE ha

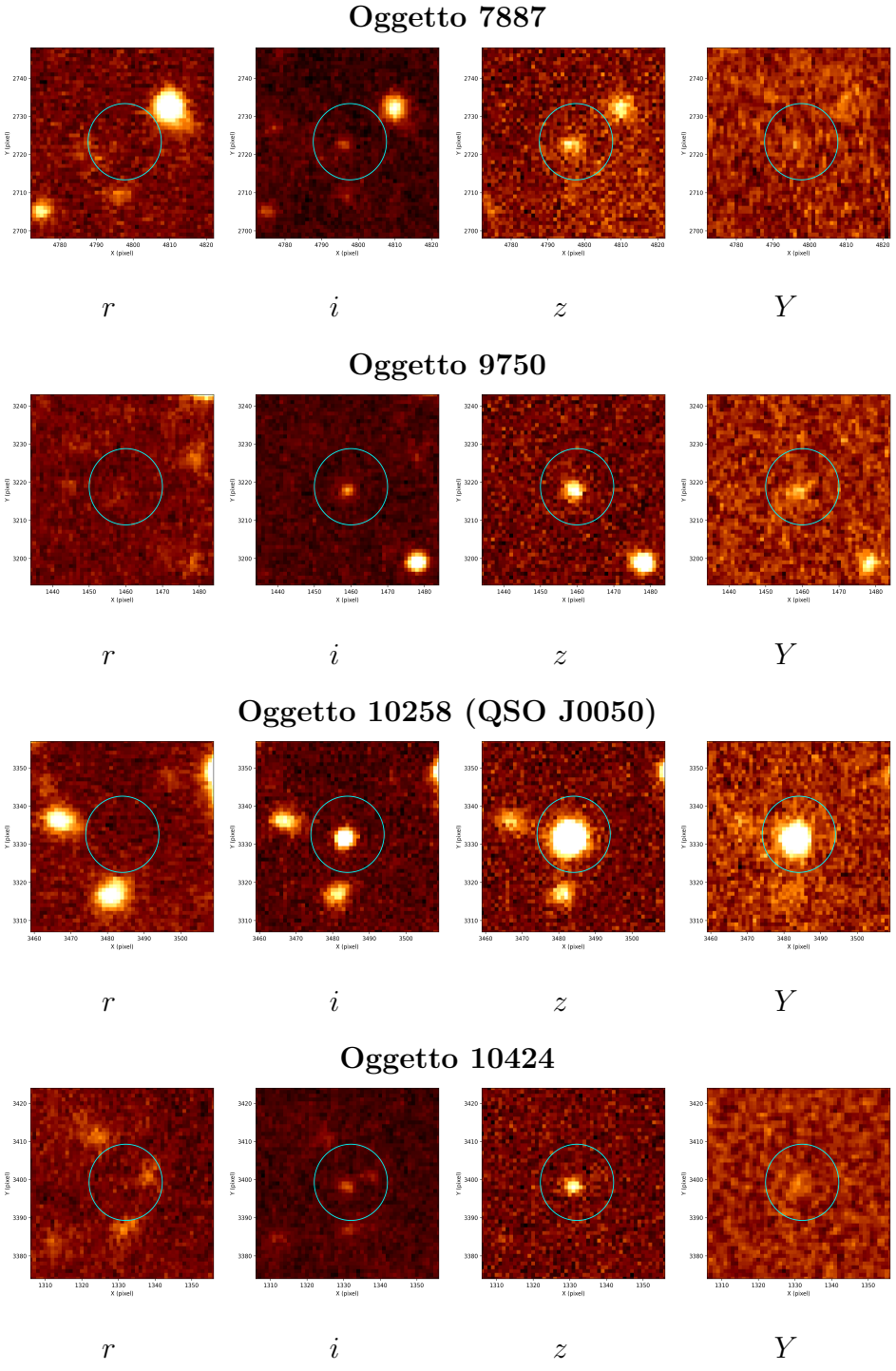


Figura 5.4: Immagini $10'' \times 10''$ in banda r, i, z e Y per i primi quattro oggetti candidati come galassie; il cerchio ha un raggio di 10 pixel, corrispondenti a ~ 2 arcsec. Per la selezione $r > mag_{2\sigma}$,ci aspettiamo che nessun oggetto venga rilevato in questa banda.

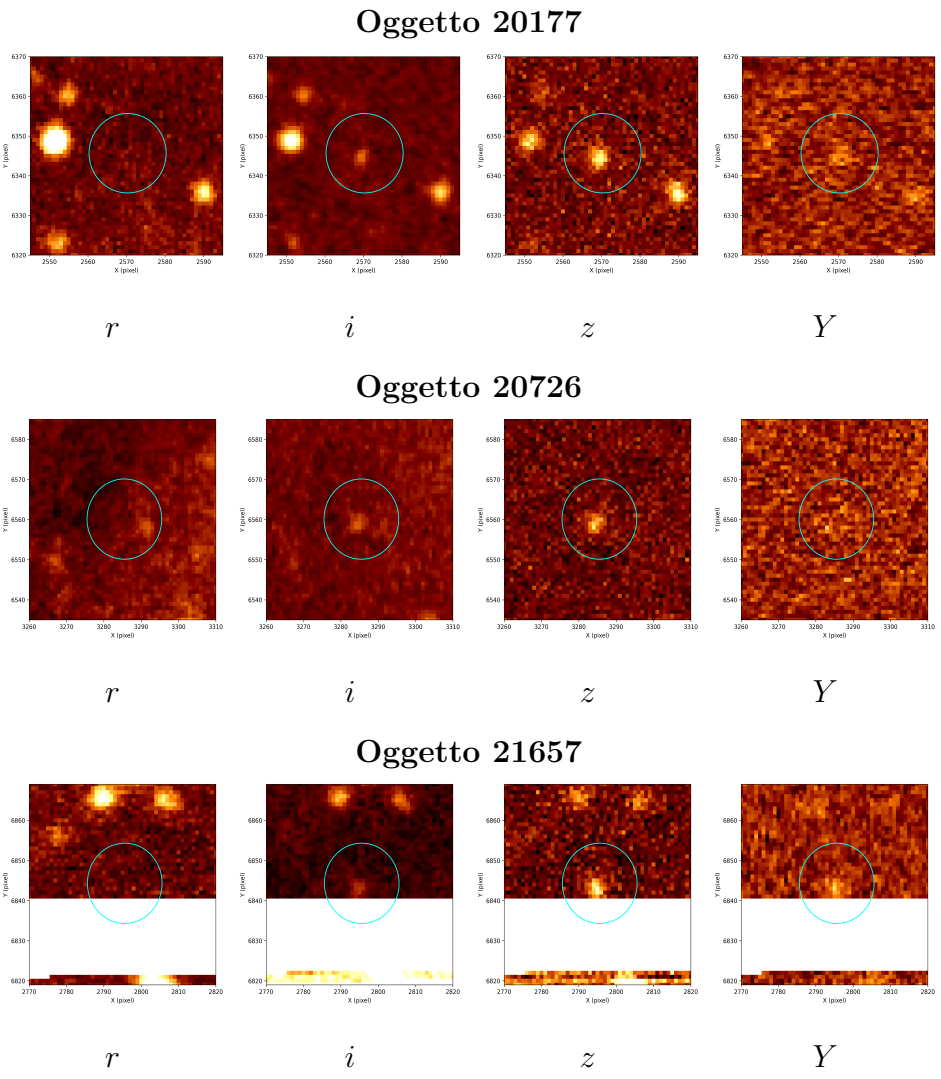
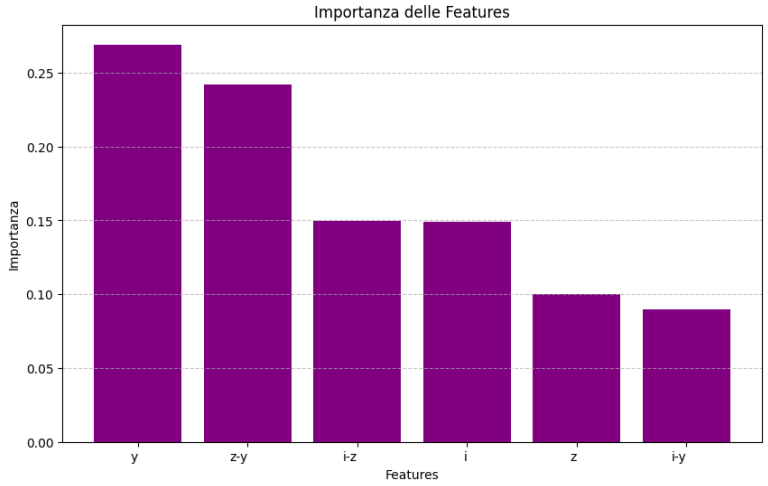


Figura 5.4: Immagini $10'' \times 10''$ in banda r, i, z e Y per i i restanti tre oggetti candidati come galassie; il cerchio ha un raggio di 10 pixel, corrispondenti a ~ 2 arcsec. Per la selezione $r > mag_{2\sigma}$,ci aspettiamo che nessun oggetto venga rilevato in questa banda.

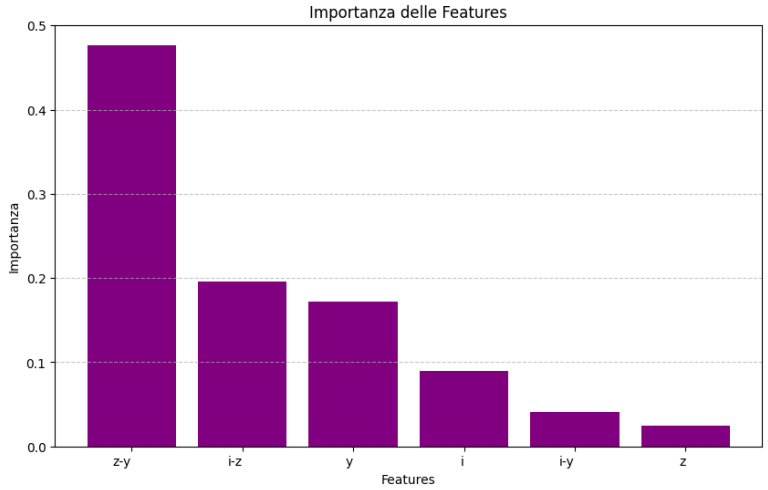
migliorato significativamente la capacità del modello di gestire la classe minoritaria, senza compromettere la performance complessiva.

Questa strategia ha permesso di ottenere una valutazione più robusta delle sette sorgenti candidate, confrontando le predizioni di modelli con approcci diversi nell'ottimizzazione della separazione tra classi e nella gestione del bilanciamento del dataset.

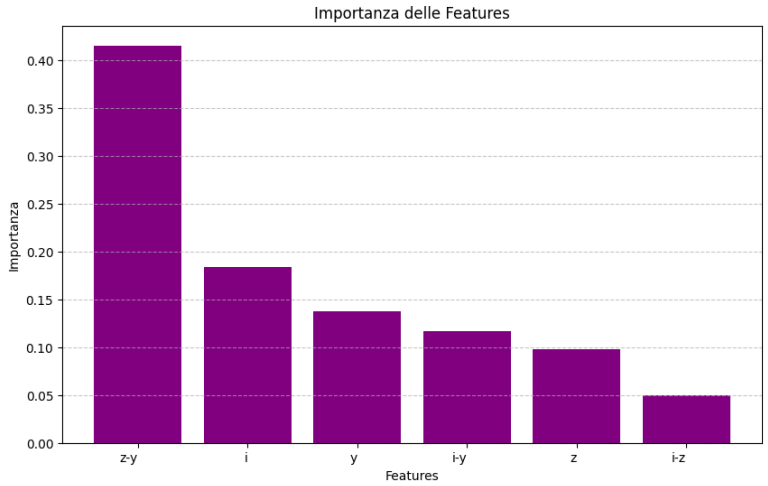
I risultati delle predizioni sono riportati nella tabella 5.3. Poiché le predizioni ottenute utilizzando i colori medi delle galassie e delle brown dwarfs coincidono, in tabella sono presentati solo tre risultati rappresentativi: RF + SMOTE, RF + LDA e PRF + SMOTE. Per sei di queste, i modelli hanno restituito un risultato coerente, classificandole tutte come galassie (G). Questa forte concordanza suggerisce che le caratteristiche fotometriche di questi oggetti rientrino



(a) PRF+SMOTE



(b) RF+LDA



(c) RF+SMOTE

Figura 5.5: Feature importances per i tre modelli: (a) PRF+SMOTE, (b) RF+LDA, (c) RF+SMOTE.

ID	RA	DEC	z ± Δz	i ± Δi	Y ± ΔY	i-z ± Δ(i-z)	z-Y ± Δ(z-Y)	i-Y ± Δ(i-Y)
7887	12.4280	34.7180	23.9 ± 0.1	26.2 ± 0.2	>24.4	2.3 ± 0.2	<-0.5	<1.7
9750	12.6817	34.7489	23.49 ± 0.07	25.1 ± 0.1	>24.4	1.6 ± 0.1	<-1.0	<0.7
10258	12.5278	34.7561	20.40 ± 0.05	23.39 ± 0.07	22.52 ± 0.09	2.99 ± 0.08	-2.1 ± 0.1	0.9 ± 0.1
10424	12.6915	34.7602	23.9 ± 0.1	25.5 ± 0.1	>24.4	1.6 ± 0.2	<-0.5	<1.1
20177	12.5973	34.9439	23.9 ± 0.1	25.9 ± 0.2	>24.4	2.0 ± 0.2	<-0.5	<1.5
20726	12.5426	34.9571	24.2 ± 0.1	25.9 ± 0.2	>24.4	1.7 ± 0.2	<-0.2	<1.5
21657	12.5799	34.9748	24.1 ± 0.1	26.4 ± 0.3	>24.4	2.3 ± 0.3	<-0.4	<1.9

Tabella 5.2: Tabella dei parametri osservativi degli oggetti selezionati.

NUMBER	**RF + SMOTE**	**RF + LDA**	**PRF + SMOTE**
7887	G	G	G
9750	G	G	G
10258	BD	G	BD
10424	G	G	G
20177	G	G	G
20726	G	G	G
21657	G	G	G

Tabella 5.3: Classificazioni ottenute per le sette sorgenti selezionate con i diversi modelli di machine learning.

chiaramente nello spazio delle feature associate alle galassie nel dataset di training, rendendo la loro classificazione robusta e indipendente dal modello utilizzato.

Tuttavia, un caso particolare emerge dall'analisi dell'oggetto 10258, il quasar attorno al quale sono state effettuate le osservazioni. Questo oggetto è stato classificato in modo discordante: RF + SMOTE e PRF + SMOTE lo identificano come brown dwarf (BD), mentre RF + LDA lo classifica come galassia (G). Questa discrepanza è interessante e merita un approfondimento.

Poiché i modelli sono stati addestrati esclusivamente su due classi—galassie e brown dwarfs—e non su quasar, è plausibile che l'algoritmo abbia cercato di assegnarlo alla categoria più simile in base alle sue caratteristiche fotometriche. I quasar ad alto redshift possono avere colori simili sia alle galassie che ai brown dwarfs, a seconda della loro emissione. L'ambiguità nella classificazione dell'oggetto 10258 potrebbe derivare dalle sue caratteristiche peculiari, che non si adattano perfettamente alle due classi considerate durante l'addestramento. In particolare, esso risulta significativamente più brillante rispetto alle galassie ed è noto che gli oggetti con un colore rosso in $(i - z)$ ma con elevata luminosità tendono a essere stelle, il che potrebbe spiegare l'incertezza nella sua classificazione.

L'analisi delle feature importance (Figura 5.5) aiuta a comprendere il motivo di questa

coerenza. Tutti e tre i modelli—PRF+SMOTE, RF+LDA e RF+SMOTE—attribuiscono una rilevanza significativa al colore $z - Y$, che risulta essere la feature più discriminante tra galassie e brown dwarfs. Il fatto che i sei oggetti siano stati classificati unanimemente come galassie indica che il loro valore di $z - Y$ rientra chiaramente nel range tipico delle galassie nel dataset di training.

Tuttavia, il peso attribuito a questa feature varia tra i modelli:

- RF+SMOTE assegna oltre il 40% dell'importanza a $z - Y$, suggerendo che questo modello si affidi quasi esclusivamente a questa informazione. Se i sei oggetti hanno valori di $z - Y$ ben distinti rispetto ai brown dwarfs, RF+SMOTE li ha classificati con alta sicurezza come galassie.

- PRF+SMOTE distribuisce l'importanza in modo più bilanciato, considerando anche la magnitudine in banda Y (con un peso del 26.9%). Questo implica che, oltre alla cromaticità, anche la luminosità assoluta ha avuto un ruolo nella classificazione, rafforzando ulteriormente l'identificazione di questi oggetti come galassie.

- RF+LDA enfatizza maggiormente le informazioni cromatiche rispetto a RF+SMOTE, con $z - Y$ che emerge come la feature dominante, rappresentando circa il 50% dell'importanza complessiva. Questo risultato suggerisce che il valore di $z - Y$ per i sei oggetti analizzati sia fortemente indicativo della classe delle galassie, determinando una classificazione coerente tra i modelli.

Un aspetto interessante è che, nonostante le differenze nell'assegnazione dei pesi alle feature, i tre modelli hanno raggiunto lo stesso risultato per questi sei oggetti. Questo suggerisce che le proprietà fotometriche delle sorgenti in questione non solo sono indicative di una galassia, ma lo sono in modo robusto rispetto a diversi approcci di classificazione. In altre parole, indipendentemente da come il modello sfrutti le informazioni disponibili, la separazione tra le due classi è netta per questi oggetti, riducendo il rischio di ambiguità.

In particolare, la banda Y gioca un ruolo chiave in PRF+SMOTE e, in misura minore, in RF+LDA, mentre è trascurata da RF+SMOTE. Questo suggerisce che le galassie non si distinguono solo per il colore $z - Y$, ma anche per la loro luminosità nelle immagini. Il fatto che tutti e tre i modelli abbiano trovato un consenso indica che sia $z - Y$ sia la banda Y sono coerenti con il comportamento atteso per una galassia e che non presentano valori al limite tra le due classi.

Al contrario, il caso dell'oggetto 10258 evidenzia un'incertezza che potrebbe derivare principalmente dalla sua luminosità, significativamente maggiore rispetto a quella degli altri candidati. Sebbene i suoi colori siano ancora più simili a quelli delle galassie rispetto agli altri oggetti, il fatto che sia così luminoso potrebbe rappresentare un'anomalia rispetto ai dati di training,

portando il modello a una classificazione meno sicura. Questo suggerisce che, oltre ai colori, anche la brillantezza dell'oggetto possa influenzare la predizione del modello. Questo spiegherebbe perché RF+SMOTE, pur assegnando un'elevata importanza a $z - Y$ (circa il 40%), ha classificato l'oggetto come brown dwarf, mentre RF+LDA, che enfatizza ancora di più questa feature (circa il 50% di importanza), lo ha invece assegnato alla classe galassia. Questo suggerisce che altri fattori, come la brillantezza dell'oggetto, potrebbero aver influenzato la decisione del modello RF+SMOTE, mentre RF+LDA, essendo più sbilanciato su $z - Y$, ha prodotto una classificazione più coerente con gli altri sei oggetti. Quindi, se da una parte il fatto che sia stato selezionato anche un QSO testimonia l'efficacia della selezione, è risaputo che le galassie a $z \sim 6$ non possono essere brillanti, il che suggerisce che la magnitudine giochi comunque un ruolo significativo nella classificazione. Allo stesso tempo, siamo alla ricerca anche di QSO, potenziali compagni del QSO centrale noto. Per questo motivo, la tecnica meno influenzata dalla luminosità potrebbe essere preferibile, poiché permette di non escludere a priori candidati interessanti.

Nel complesso, il consenso tra i modelli per le sei galassie conferma che l'addestramento ha portato a una classificazione affidabile per gli oggetti con caratteristiche chiaramente distinguibili. Tuttavia, il caso 10258 dimostra che, quando un oggetto ha valori fotometrici più ambigui, la dipendenza di alcuni modelli da una singola feature può portare a discrepanze nei risultati.

Infine, questi risultati evidenziano l'importanza di un approccio che bilanci l'uso delle feature, evitando di affidarsi eccessivamente a una singola variabile. PRF+SMOTE, avendo una distribuzione più equilibrata delle feature importance, sembra offrire una maggiore stabilità rispetto a RF+SMOTE e a RF+LDA, che potrebbero risultare più sensibili alle incertezze nei dati.

Conclusioni e sviluppi futuri

In questa tesi è stato esplorato l'uso di algoritmi di machine learning come metodo alternativo per l'identificazione delle galassie Lyman-break (LBGs) nelle vicinanze (~ 10 Mpc) di quasar ad alto redshift ($z \gtrsim 6$). L'analisi si è basata su osservazioni fotometriche ottiche e nel vicino infrarosso, con l'obiettivo di studiare l'ambiente dei quasar, la relazione tra i SMBHs e il loro contesto cosmico, e l'impatto del feedback dei quasar sulla formazione delle galassie circostanti. In particolare, l'identificazione delle LBGs a redshift $z \sim 6$ risulta complicata a causa delle loro magnitudini molto deboli e della presenza di sorgenti astrofisiche a redshift più bassi che simulano le caratteristiche fotometriche delle galassie ad alto redshift utilizzate per identificarle. Le contaminanti più numerose risultano essere le stelle nane di tipo M, L e T nella nostra galassia, a causa dei loro colori molto rossi che possono coincidere con quelli delle LBGs candidate ad alto redshift e che possono presentare una morfologia puntiforme simile agli oggetti compatti ad altissimo redshift.

Questa tesi ha dimostrato come gli algoritmi di machine learning possano essere efficaci nell'identificazione di galassie Lyman-break (LBG) attorno a quasar a $z > 6$, migliorando la separazione tra galassie ad alto redshift e contaminanti, come le brown dwarfs, grazie all'integrazione delle incertezze osservative. Il modello sviluppato sarà applicato ai dati del progetto SQUEEzE, che osserva 15 campi di quasar a $z \sim 6$ con imaging multi-banda (r, i, z e Y) fino alla magnitudine $z \sim 25 - 25.5$.

I dati utilizzati per l'addestramento e la validazione dei modelli provengono dal catalogo SHELLQs, che raccoglie osservazioni della Hyper Suprime-Cam (HSC) sul telescopio Subaru per oggetti a $5.7 < z < 6.9$, mentre il test set si basa sui dati di Balmaverde et al. (2017) per il campo attorno al quasar SDSSJ1030+0524 ($z = 6.28$), contenente 16 galassie e 16 brown dwarfs. Le galassie nel campo J1030 sono più deboli, con magnitudini più grandi di circa $\sim 1 - 2$ mag rispetto a quelle del dataset SHELLQs, e presentano numerosi upper limits nelle bande i e Y, mentre le brown dwarfs mostrano distribuzioni fotometriche simili nei due dataset. L'applicazione di un adeguato preprocessing ha migliorato l'adattamento del modello ai dati e la valutazione delle sue performance.

L'analisi delle incertezze nelle misurazioni astronomiche è stata affrontata con due approcci

principali. Il primo è stato il metodo Monte Carlo, che ha permesso di generare campioni sintetici seguendo una distribuzione normale basata sulla magnitudine osservata e il suo errore associato, migliorando così la robustezza del modello e riducendo la sensibilità agli errori strumentali. Il secondo approccio è stato l'uso della Probabilistic Random Forest, che ha incorporato direttamente le incertezze nel processo di classificazione senza necessità di simulazioni aggiuntive, ottimizzando il costo computazionale.

Una delle sfide più complesse ha riguardato la gestione delle magnitudini non rilevate. Sono stati testati due metodi:

- l'LLM utilizza il limite inferiore come media di una distribuzione gaussiana troncata a tale valore. Questo approccio garantisce il rispetto dei vincoli fisici dei dati, ma potrebbe introdurre un bias nella stima delle magnitudini, sottostimandone la reale distribuzione.
- Il CBE utilizza il colore medio della classe per stimare le magnitudini non rilevate, generando poi campioni da una distribuzione normale troncata al limite inferiore, con media basata sulla stima e deviazione standard adattata ai limiti osservativi. Questo metodo bilancia semplicità e realismo, evitando troncature arbitrarie e preservando una maggiore variabilità nei dati simulati.

I risultati mostrano che il trattamento delle magnitudini non rilevate influisce sulle prestazioni del modello: adottare l'approccio LLM garantisce buoni risultati (F1-score $\sim$93% e accuracy $\sim$91-93%), mentre l'approccio CBE porta a un calo di accuratezza (fino all'84% nel test set). Questo calo è dovuto al fatto che le magnitudini stimate con il metodo CBE risultano più realistiche, riflettendo meglio la natura degli oggetti di J1030, che sono generalmente più deboli. Di conseguenza, la differenza tra il dataset di training SHELLQs e quello di test J1030 diventa più marcata, rendendo più difficile per il modello generalizzare correttamente da un set di dati all'altro. L'uso di SMOTE non sempre migliora le prestazioni, risultando utile in particolare quando si adotta l'approccio CBE (+4% di accuracy per il test set). La LDA si dimostra efficace nel stabilizzare le prestazioni (accuracy $\sim$94%), ma il miglior risultato si ottiene con la Probabilistic Random Forest (PRF), che gestisce meglio le incertezze e, combinata con SMOTE, raggiunge il massimo F1-score (97%) e accuracy (97%) nel test set.

In definitiva, i risultati evidenziano che la scelta del metodo dipende da diversi fattori e, in particolare, dal modo in cui vengono trattate le incertezze, le magnitudini non rilevate e, di conseguenza, i colori che presentano limiti inferiori/superiori. Se si preferisce un approccio più conservativo, il modello Random Forest con LDA senza SMOTE offre prestazioni stabili e affidabili, con un'accuracy del 94%. Se invece si vuole sfruttare al massimo l'informazione disponibile, stimando le magnitudini non rilevate e bilanciando il dataset con SMOTE, allora PRF si conferma la scelta migliore, raggiungendo un'accuracy del 97% nel test set. Per una

valutazione più completa, è stato testato anche Random Forest con SMOTE, offrendo un confronto tra diverse strategie di bilanciamento e gestione delle incertezze.

Questi tre modelli – Random Forest con LDA, Random Forest con SMOTE e Probabilistic Random Forest con SMOTE – sono stati quindi applicati per classificare gli oggetti nel campo del quasar J0050 a $z = 6.246$ pre-selezionati tramite criteri di colori ottici/IR, in linea con l'obiettivo della tesi di sviluppare modelli di machine learning per l'identificazione di galassie ad alto redshift. La pre-selezione ha portato ad un dataset di sette oggetti, tra cui il quasar J0050, noto a priori. I risultati mostrano una forte coerenza per sei sorgenti, classificate unanimemente come galassie da tutti i modelli, a conferma della loro robustezza. Tuttavia, è emerso un caso particolare: l'oggetto 10258, corrispondente al quasar J0050, è stato classificato in modo discordante. Poiché il training set conteneva solo galassie e brown dwarfs, i modelli hanno probabilmente assegnato il quasar alla classe più simile in base ai colori e alla luminosità. L'analisi delle feature importance ha evidenziato il ruolo cruciale del colore $z - Y$ nella classificazione, suggerendo che una migliore caratterizzazione delle sorgenti potrebbe affinare ulteriormente le prestazioni del modello.

I risultati ottenuti aprono diverse prospettive per migliorare ulteriormente la classificazione delle galassie Lyman-break a $z > 6$. Un possibile primo passo potrebbe essere l'inclusione di un campione di quasar nel set di addestramento, ad esempio selezionando un gruppo di QSOs dal catalogo SHELLQs. Questo permetterebbe al modello di riconoscere questa classe, riducendo possibili ambiguità e migliorando la capacità di generalizzazione.

Un altro sviluppo importante riguarda l'integrazione del dataset J1030 nel training set. Attualmente, infatti, il dataset di addestramento è composto prevalentemente da oggetti più brillanti rispetto a quelli ricercati. Questa differenza di magnitudine potrebbe influenzare la capacità del modello di generalizzare correttamente, soprattutto per gli oggetti che si trovano al limite tra le due classi. L'aggiunta di J1030 consentirebbe al modello di adattarsi meglio alle caratteristiche fotometriche di questa popolazione, migliorando la sua capacità di distinguere tra galassie e brown dwarf.

Oltre all'ampliamento del training set, un'altra strategia utile potrebbe essere quella di includere nuove caratteristiche nel dataset, come l'estensione spaziale degli oggetti. Questa informazione potrebbe fornire un ulteriore criterio di separazione tra le classi, affinando la classificazione e contribuendo a una migliore interpretazione fisica dei risultati. In particolare, questo parametro potrebbe essere ricavato attraverso un'analisi morfologica basata sul classico diagramma mag(aperture) vs. mag(total), in quanto il rapporto tra queste due quantità rappresenta un indicatore robusto della concentrazione della sorgente (Balmaverde et al. 2017).

Tuttavia, tale caratteristica non è stata inclusa in questa fase, poiché nella fase preliminare ci siamo concentrati esclusivamente sui dati disponibili, e il dataset SHELLQs utilizzato per addestrare gli algoritmi non includeva queste informazioni.

Parallelamente, sarebbe interessante applicare i modelli sviluppati anche agli altri campi del progetto SQUEEzE. Finora, le predizioni sono state effettuate su un solo campo, ma estenderle agli altri permetterebbe di esplorare sistematicamente l'intero dataset e identificare potenziali candidati di interesse. Sebbene non sia possibile verificare direttamente l'accuratezza delle classificazioni senza un follow-up spettroscopico, confrontare le distribuzioni delle predizioni tra i vari campi potrebbe fornire indicazioni sulla coerenza del modello e su eventuali anomalie nei risultati.

Una volta completato il follow-up spettroscopico sui campi di SQUEEzE, sarebbe opportuno includere anche questi nuovi oggetti nel training set. Infatti, come detto in precedenza, un training set più rappresentativo, che includa anche sorgenti più deboli rispetto a quelle presenti in SHELLQs, contribuirebbe a migliorare la robustezza della classificazione e a ridurre ulteriormente le incertezze nei risultati. L'integrazione di questi nuovi dati renderebbe il modello più completo e meglio adattato alla popolazione osservata, aumentando la sua affidabilità nelle analisi future. Inoltre, i campioni di SQUEEzE, che osservano quasar a $z \sim 6$, presentano caratteristiche simili a quelle degli oggetti nel campo di J1030, che, come già sottolineato, mostrano magnitudini più deboli e numerosi upper limits nelle bande i e Y, rispetto agli oggetti di SHELLQs. Queste stesse condizioni sono riscontrabili anche nei campioni di SQUEEzE, rendendo l'inclusione di questi dati particolarmente vantaggiosa per perfezionare il modello. Va inoltre considerato che il dataset J1030, contenente solo circa 30 oggetti, è troppo limitato per contribuire in modo significativo al training set. L'inclusione dei dati di SQUEEzE, che racchiudono sorgenti più simili a quelle di J1030, rappresenterebbe quindi un passo cruciale per migliorare ulteriormente la classificazione, aumentando la capacità del modello di distinguere tra le diverse classi e riducendo l'impatto delle incertezze nei risultati.

Infine, si potrebbero esplorare altre metodologie di machine learning, confrontandole con quelle applicate fino ad ora. In particolare, algoritmi come Gradient Boosting (ad esempio XGBoost e LightGBM) e Reti Neurali (come Multilayer Perceptron) potrebbero offrire vantaggi in termini di prestazioni.

- Gradient Boosting (XGBoost e LightGBM): questo approccio costruisce una serie di alberi decisionali uno alla volta, in cui ogni nuovo albero cerca di correggere gli errori fatti dal precedente. Rispetto al Random Forest, che costruisce alberi indipendenti tra loro, il Gradient Boosting cerca di migliorare progressivamente il modello, mettendo maggiore enfasi sugli errori commessi e migliorando le previsioni in modo più preciso. Questo rende

il Gradient Boosting più adatto per casi in cui si desidera ottenere un modello molto potente e preciso, ma richiede più tempo per allenarsi e maggiore attenzione per evitare che il modello si adatti troppo ai dati (cioè sovra-allenamento).

- Reti Neurali (MLP): le Reti Neurali sono in grado di catturare pattern complessi nei dati. In particolare, il Multilayer Perceptron (MLP) è una rete semplice che può essere utilizzata per classificare i dati astronomici, anche se, a differenza di altri algoritmi, richiede generalmente un numero maggiore di dati per funzionare al meglio.

Confrontando questi modelli con quelli già utilizzati, sarà possibile valutare eventuali miglioramenti nelle prestazioni, affinando ulteriormente la capacità di classificazione e garantendo una maggiore affidabilità nei risultati.

In sintesi, il lavoro svolto in questa tesi rappresenta un primo passo verso l'uso sistematico del machine learning per identificare galassie Lyman-Break attorno ai quasar ad alto redshift. Le future evoluzioni del modello, con dataset più ampi, nuove metodologie di classificazione e un migliore trattamento delle incertezze, potranno affinare ulteriormente questa tecnica e contribuire a una più completa comprensione dell'ambiente dei quasar nell'Universo primordiale.

Appendice A

Vocabolario dei termini tecnici

Algoritmi di Ensemble:
Gli algoritmi di ensemble sono tecniche di apprendimento automatico che combinano le previsioni di più modelli di base (solitamente deboli) per migliorare le performance complessive rispetto a un singolo modello. I metodi di ensemble, come il *Random Forest* e il *Boosting*, utilizzano la diversità tra i modelli per ridurre il rischio di overfitting e aumentare la capacità di generalizzazione, ottenendo un modello finale che risulta più robusto e accurato.

Bias:
Il bias rappresenta l'errore sistematico introdotto dall'approssimazione del modello rispetto alla complessità del problema reale. Un alto bias è tipico di modelli semplici, che non riescono a catturare sufficientemente le relazioni tra caratteristiche e classi nei dati, portando a previsioni inaccurate sia sui dati di training sia su quelli di test. Questo fenomeno è noto come *underfitting* e si verifica quando il modello non ha capacità sufficiente per rappresentare la struttura del dataset.

Caratteristiche:
Nell'ambito del machine learning, le caratteristiche (o features) sono variabili o attributi che rappresentano le informazioni utilizzate da un modello per apprendere. Possono essere misurazioni, valori numerici, categoriali o trasformazioni dei dati grezzi, e sono fondamentali per descrivere i dati in modo utile per il compito di apprendimento.

Classe:
Categoria o etichetta assegnata a un oggetto in un problema di classificazione. Le classi rappresentano le possibili categorie di appartenenza degli oggetti, come galassie, quasar o brown dwarf nel contesto astrofisico studiato in questo lavoro di tesi.

Dati:

L'insieme completo di oggetti e delle loro caratteristiche in un dataset. I dati rappresentano le osservazioni raccolte, incluse le informazioni relative alle classi e alle feature associate. Esempio: I dati del catalogo SHELLQs includono oggetti astronomici descritti da magnitudini, errori associati e classi di appartenenza.

Dati rumorosi:
Dati che contengono variabilità o errori indesiderati, spesso causati da imprecisioni strumentali, errori di misura, o fenomeni aleatori non rilevanti per l'analisi. I dati rumorosi possono rendere più difficile per i modelli identificare pattern significativi o fare previsioni accurate, poiché il rumore può mascherare le informazioni utili contenute nei dati stessi.

Dimensionalità:
Il numero di caratteristiche o variabili che definiscono uno spazio in cui i dati sono rappresentati. Ad esempio, in un dataset con tre caratteristiche (x, y, z), i dati sono rappresentati in uno spazio tridimensionale. La riduzione della dimensionalità è una tecnica utilizzata per semplificare i modelli, riducendo il numero di caratteristiche mantenendo la maggior quantità possibile di informazioni rilevanti.

Entropia:
derivata dalla teoria dell'informazione, l'entropia misura il grado di disordine o incertezza all'interno di un nodo, cioè quanto è imprevedibile o mista la distribuzione delle classi all'interno di un nodo. Per una distribuzione di probabilità delle classi p_j nel nodo, l'entropia è calcolata come:

$$E = -\sum_{j=1}^{C} p_j \log_2(p_j)$$

Un nodo classificato in una sola classe avrà entropia zero (massima purezza), mentre un nodo con una distribuzione uniforme tra le classi avrà l'entropia massima ($log_2(C)$). Questo criterio tiene conto non solo della distribuzione delle classi, ma anche della frequenza relativa delle stesse. In particolare, penalizza maggiormente i nodi in cui le classi sono distribuite in modo uniforme, poiché l'incertezza è massima quando tutte le classi sono ugualmente rappresentate. Inoltre, se una classe è molto rara, la sua presenza nel nodo aumenta l'incertezza, ma l'impatto sull'entropia è inferiore rispetto a quello di una classe più frequente. Immaginiamo un nodo contenente 10 oggetti astronomici, suddivisi in classi come segue:

- 5 Galassie
- 3 Stelle
- 2 Quasar

L'entropia del nodo sarà calcolata come:

$$E = -\left(\frac{5}{10}\log_2\frac{5}{10}\right) - \left(\frac{3}{10}\log_2\frac{3}{10}\right) - \left(\frac{2}{10}\log_2\frac{2}{10}\right) \approx 1.485$$

Il valore di entropia derivato, 1.485, indica una distribuzione mista del nodo. Se successivamente dividiamo il nodo in due rami, uno contenente solo Galassie (entropia $E = 0$) e l'altro contenente Stelle e Quasar in proporzione 3:2, l'entropia di quest'ultimo sarà $E \approx 0.971$. Questa suddivisione riduce l'entropia complessiva rispetto al nodo originale, portando a una maggiore purezza.

Indice di Gini:
misura la probabilità che due oggetti scelti casualmente da un insieme appartengano a classi diverse. Questo rappresenta uno dei criteri di impurità più utilizzati negli alberi di classificazione. Se p_j è la probabilità di appartenenza di un oggetto in un nodo della classe j e C è il numero di classi, l'indice di Gini è definito come:

$$G = 1 - \sum_{j=1}^{C} p_j^2$$

Questo valore sarà minimo (0) quando gli oggetti nel nodo appartengono tutti alla stessa classe (massima purezza) e massimo (1) quando le classi sono equamente distribuite. Ad esempio, immaginiamo un nodo contenente 10 oggetti astronomici, con la seguente distribuzione:

- 5 Galassie
- 3 Stelle
- 2 Quasar

L'indice di Gini sarà calcolato come:

$$G = 1 - \left(\frac{5}{10}\right)^2 - \left(\frac{3}{10}\right)^2 - \left(\frac{2}{10}\right)^2 = 0.62$$

Se il nodo successivo contiene solo Galassie, l'indice di Gini sarà $G = 0$ indicando massima purezza.

Modello di apprendimento:
Un modello di apprendimento è una funzione matematica o statistica che, una volta che vengono forniti i dati di input (caratteristiche), è in grado di fare previsioni o classificazioni.

Modello di apprendimento supervisionato:
Il modello di apprendimento supervisionato è un approccio dell'apprendimento automatico in cui un modello viene addestrato su un set di dati forniti dall'utente in cui ogni campione è

associato a una classe. Durante il processo di addestramento, il modello impara a correlare le caratteristiche di input con le classi, così da poter effettuare previsioni su nuovi dati non classificati.

Oggetto:
Un'unità individuale del dataset che viene analizzata o classificata. Ogni oggetto è descritto da un insieme di caratteristiche (*feature*) e può appartenere a una specifica classe. Ad esempio: Un oggetto può essere un'entità astronomica come una galassia con magnitudini specifiche nelle bande i, y, z.

Overfitting:
Si verifica quando un modello di apprendimento automatico impara troppo bene i dettagli e il rumore dei dati di addestramento, al punto da adattarsi eccessivamente a questi specifici dati. Di conseguenza, il modello ottiene ottimi risultati sui dati su cui è stato addestrato, ma fatica a generalizzare e a fare previsioni corrette su nuovi dati, riducendo la sua capacità di essere utile in situazioni reali.

Riduzione di impurità:
La riduzione di impurità è una misura utilizzata negli alberi decisionali per quantificare il miglioramento ottenuto suddividendo un nodo in due sotto-nodi. L'obiettivo è massimizzare questa riduzione, in modo da ottenere partizioni che separino meglio le classi o i valori target. Matematicamente, la riduzione di impurità è definita come:

$$\Delta I = I_{parent} - (\frac{N_{left}}{N_{total}} I_{left} + \frac{M_{right}}{N_{total}} I_{right}) \tag{A.1}$$

Dove:

- I_{parent} è l'impurità del nodo padre.
- I_{left}: è l'impurità del nodo figlio sinistro.
- I_{right} è l'impurità del nodo figlio destro.
- N_{total} è il numero totale di oggetti nel nodo padre.
- N_{left} è il numero di oggetti nel nodo figlio sinistro.
- N_{right}è il numero di oggetti nel nodo figlio destro.

.

Spazi a bassa dimensione:
Rappresentazioni degli oggetti in cui il numero di caratteristiche (dimensioni) è ridotto rispetto

allo spazio originario. Ad esempio, se un dataset ha 100 caratteristiche, ma viene proiettato in uno spazio con 2 o 3 caratteristiche principali, si dice che i dati sono stati ridotti a uno spazio a bassa dimensione. Questa riduzione è spesso utilizzata per semplificare l'analisi, ridurre il rumore o visualizzare i dati in modo più intuitivo.

Varianza:
La varianza misura quanto un modello cambia le sue predizioni al variare dei dati di training. Un'alta varianza indica che il modello è eccessivamente sensibile ai cambiamenti nei dati, adattandosi troppo strettamente alle specificità (e al rumore) del set di training. Questo fenomeno è noto come overfitting e può portare a una scarsa capacità di generalizzazione del modello su dati nuovi. La varianza è generalmente elevata nei modelli molto complessi, come quelli con molte caratteristiche o gradi di libertà.

Appendice B

Test di Box M per l'Omoschedasticità

Il **Test di Box M** è un test statistico utilizzato per verificare l'assunzione di omogeneità delle varianze e covarianze tra più gruppi, un requisito importante nell'analisi discriminante lineare (LDA). Questo test è utile quando si analizzano più gruppi e si vuole determinare se le varianze e le covarianze tra i gruppi sono simili, ovvero se possiamo applicare l'LDA o se è necessario ricorrere all'analisi discriminante quadratica (QDA).

B.1 Descrizione del test di Box M

Il **test di Box M** verifica se le matrici di covarianza di più gruppi siano omogenee (uguali). In altre parole, testiamo se la dispersione delle variabili (le varianze e covarianze) è simile per tutti i gruppi considerati. Se il test di Box M restituisce un risultato significativo (p-value basso) significa che le covarianze tra i gruppi sono significativamente diverse, e quindi l'assunzione di omogeneità non è soddisfatta.

Matematicamente, il test di Box M si basa sulla statistica:

$$M = \frac{n-k}{(k-1)ln|\sum_{pooled}|} \sum_{i=1}^{k} [(n_i - 1)ln|\sum_i|]$$

Dove:

- n è il numero totale di osservazioni nel campione.
- k è il numero di gruppi.
- n_i è il numero di osservazioni nel gruppo i.
- $\sum_i$ è la matrice di covarianza del gruppo i.
- $\sum_{pooled}$ è la matrice di covarianza combinata di tutti i gruppi.

La statistica M segue la distribuzione del chi-quadro (χ^2) con un numero di gradi di libertà che dipende dal numero di variabili e dal numero di gruppi.

B.2 Applicazione del Test di Box M nel contesto dell'analisi discriminante

Nel contesto dell'LDA, l'assunzione di **omoschedasticità** (uguali varianze e covarianze per tutti i gruppi) è cruciale per il corretto funzionamento dell'algoritmo. Se il test di Box M indica che questa assunzione è violata, si dovrebbe passare alla QDA, che non richiede questa condizione. La QDA permette di trattare ciascun gruppo come avente la propria matrice di covarianza, consentendo una maggiore flessibilità, ma sacrificando la semplicità del modello lineare dell'LDA.

B.3 Interpretazione del Test di Box M

Il p-value derivante dal test di Box M indica se le matrici di covarianza sono significativamente differenti tra i gruppi. In generale:

- se **p-value** > 0.05, non possiamo rifiutare l'ipotesi nulla, e possiamo assumere che le varianze e covarianze siano omogenee tra i gruppi. In questo caso, l'LDA è appropriato.
- Se **p-value** < 0.05, rifiutiamo l'ipotesi nulla, indicando che le varianze e covarianze sono significativamente differenti tra i gruppi. In tal caso, l'analisi discriminante quadratica (QDA) è una scelta migliore rispetto all'LDA, poiché la QDA non assume l'omoschedasticità.

B.4 Esempio pratico

Supponiamo di voler analizzare un dataset con tre classi di oggetti astronomici: Galassie, Quasar e Brown Dwarf, utilizzando due caratteristiche ($i - z$ e $z - y$). Per applicare l'LDA, dobbiamo verificare che le varianze e covarianze di queste due caratteristiche siano simili per tutte e tre le classi.

Applicando il test di Box M, otteniamo un p-value di 0.03. Poiché il p-value è inferiore a 0.05, possiamo rifiutare l'ipotesi nulla e concludere che le covarianze non sono omogenee tra le classi. Pertanto, dovremo utilizzare la QDA anziché l'LDA.

B.5 Limiti del test box M

Il test di Box M ha alcune limitazioni:

- È sensibile alla deviazione dalla normalità dei dati. Se i dati non seguono una distribuzione normale multivariata, il test potrebbe non essere affidabile.
- Può essere influenzato dalle dimensioni dei campioni. Se il numero di osservazioni nei gruppi è molto sbilanciato, il test potrebbe non funzionare correttamente.
- Il test diventa meno efficace con un numero elevato di variabili (dimensione dello spazio delle caratteristiche) rispetto al numero di osservazioni.

B.6 Conclusioni

Il test di Box M è un utile strumento statistico per verificare l'assunzione di omogeneità delle covarianze, fondamentale per l'applicazione dell'LDA. Questo test consente di valutare se le matrici di covarianza dei gruppi sono uguali, un presupposto chiave per la corretta implementazione dell'LDA. Tuttavia, quando le covarianze non sono uguali tra i gruppi, è consigliabile utilizzare la QDA per ottenere modelli più robusti e precisi. In ogni caso, è importante considerare i limiti del test, in particolare la sua sensibilità alla non normalità dei dati.

Nel nostro caso, il test di Box M è stato applicato al dataset in analisi, restituendo un p-value di 1. Questo risultato indica l'assenza di differenze significative tra le matrici di covarianza dei gruppi, permettendoci di procedere con l'applicazione dell'LDA con fiducia nella validità dell'assunzione di omogeneità.

Appendice C

Codice

Listing C.1: Librerie

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, confusion_matrix,
    classification_report
import seaborn as sn
import os
import pickle
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from scipy.stats import truncnorm
from PRF import prf
from collections import Counter
```

Listing C.2: Funzioni

```

def evaluate(model, X_test, y_test, features, path, val_or_test):
    """Valuta un modello di classificazione e salva i risultati, tra cui il
    modello stesso, la matrice di confusione e il classification report.
    model: il modello di classificazione addestrato.
    X_test: il dataset di test (features).
    y_test: le etichette di test.
    features: lista delle feature utilizzate nel modello.
    path: directory in cui salvare i risultati.
    val_or_test: stringa che indica se il set di dati e' di validazione (val
    ) o test (test).
    """
```

```
11
12      # Salva il modello in un file
13      pickle.dump(model, open(f"{path}/model.sav", 'wb'))
14
15      # Predizioni del modello
16      y_pred = model.predict(X_test)
17
18      # Calcola l'accuratezza
19      accuracy = accuracy_score(y_test, y_pred)
20
21      # Calcola la matrice di confusione
22      labels = np.unique(y_test)
23      cm = confusion_matrix(y_test, y_pred, labels=labels)
24
25      # Visualizza la matrice di confusione con una heatmap
26      plt.figure()
27      ax = plt.subplot()
28      conf_matrix = sn.heatmap(cm, annot=True, ax=ax)
29      ax.set_xlabel("Predicted")
30      ax.set_ylabel("True")
31      ax.set_title('Confusion Matrix')
32      ax.xaxis.set_ticklabels(labels)
33      ax.yaxis.set_ticklabels(labels)
34      plt.savefig(f"{path}/confusion_matrix_{val_or_test}.png")
35      plt.clf()
36
37      # Genera e salva il classification report
38      report = classification_report(y_test, y_pred, target_names=labels,
    output_dict=True)
39      report_df = pd.DataFrame(report).transpose()
40      report_df.to_csv(f"{path}/classification_report_{val_or_test}.csv")
41
42      return accuracy
43
44  def generate_samples(df, sigma, n_samples, use_lower_limits=False):
45      """Questa funzione genera campioni Monte Carlo per incorporare le
    incertezze nei dati osservativi e trattare i valori censurati (limiti
    superiori/inferiori nelle magnitudini).
46      Parametri:
47      - df: (DataFrame) Il dataset originale contenente le magnitudini (i, y,
    z), gli errori associati (i_err, y_err, z_err), e le colonne che indicano
     se il valore e' censurato (i_soglia, y_soglia, z_soglia).
48      - sigma: (int) Definisce la deviazione standard utilizzata per la
    distribuzione troncata quando si generano valori Monte Carlo per le
    magnitudini censurate.
```

```
    - n_samples: (int) Numero di campioni da generare per ogni oggetto del dataset.
    - use_lower_limits: (bool, opzionale) Se True, utilizza i limiti inferiori invece di stimare il valore censurato dai colori medi delle classi.
"""
    new_data = []  # Lista per raccogliere i nuovi campioni generati

    # Calcolare i colori medi per classe utilizzando solo i valori NON censurati
    # I colori vengono calcolati come differenza tra le magnitudini: (i-z), (z-y), (i-y)
    color_means = {}
    for classe in df['Target'].unique():
        subset = df[(df['Target'] == classe) & (df['i_soglia'] == 0) & (df['y_soglia'] == 0) & (df['z_soglia'] == 0)]
        if not subset.empty:
            color_means[classe] = {
                'i-z': np.mean(subset['i'] - subset['z']),
                'z-y': np.mean(subset['z'] - subset['y']),
                'i-y': np.mean(subset['i'] - subset['y'])
            }

    # Calcolare la media degli errori per classe
    error_means = {}
    for classe in df['Target'].unique():
        subset = df[df['Target'] == classe]
        error_means[classe] = {
            'i_err': np.mean(subset['i_err']),
            'y_err': np.mean(subset['y_err']),
            'z_err': np.mean(subset['z_err'])
        }

    #  Per ogni riga nel dataset originale, generiamo n_samples campioni Monte Carlo
    for idx, row in df.iterrows():
        classe = row['Target']
        color_mean = color_means.get(classe, {'i-z': 0, 'z-y': 0, 'i-y': 0})  # Colori stimati per la classe
        error_mean = error_means.get(classe, {'i_err': 0, 'y_err': 0, 'z_err': 0})  # Errori medi per la classe

        for i in range(n_samples):  # Generiamo n_samples nuovi per ogni oggetto
            new_row = {'Target': row['Target'], 'Name': row['Name']}  #
```

```
    Nuova riga di dati
84
85          # Campionamento delle magnitudini con gestione degli errori
86          for mag, mag_err, mag_soglia in [('i', 'i_err', 'i_soglia'),
87                                           ('y', 'y_err', 'y_soglia'),
88                                           ('z', 'z_err', 'z_soglia')]:
89
90              if row[mag_soglia] == 0:  # Non censurato -> Usare il valore
     originale
91                  new_row[mag] = row[mag]
92                  new_row[mag_err] = row[mag_err]
93              else:  # Censurato -> Stima della magnitudine
94                  if mag == 'i':
95                      new_row[mag] = row['z'] + color_mean['i-z']  # Stima
     di i usando i-z
96                  elif mag == 'z':
97                      new_row[mag] = row['y'] + color_mean['z-y']  # Stima
     di z usando z-y
98                  else:
99                      new_row[mag] = row['i'] - color_mean['i-y']  # Stima
     di y usando i-y
100
101                 lower_bound = row[mag]  # Il valore censurato e' un
    limite superiore
102                 mean = new_row[mag]  # Media stimata della distribuzione
103                 std_dev = abs((lower_bound - new_row[mag]) / sigma)  #
    Deviazione standard stimata
104
105                 if use_lower_limits:
106                     mean = row[mag]  # Se vogliamo usare il limite
    inferiore
107                     std_dev = error_mean.get(f'{mag}_err', sigma)  # Usa
     la deviazione standard media della classe
108
109                 # Generazione di un valore casuale da una distribuzione
    troncata
110                 a, b = (lower_bound - mean) / std_dev, np.inf  #
    Troncatura a sinistra (upper limit)
111                 sample = truncnorm.rvs(a, b, loc=mean, scale=std_dev,
    size=1)  # Estrazione del valore
112
113                 new_row[mag] = sample[0]  # Assegna il valore campionato
114                 new_row[mag_err] = std_dev  # Registra la deviazione
    standard usata
115
```

```
116             # Aggiungi la nuova riga con i campioni generati alla lista
117             new_data.append(new_row)
118 
119     # Creazione del DataFrame finale con i campioni generati
120     columns = ['Name', 'Target', 'i', 'y', 'z', 'i_err', 'y_err', 'z_err']
121     new_df = pd.DataFrame(new_data, columns=columns)
122 
123     return new_df
124 
125 def preprocessing(file_path, dataset_name, sigma, n_samples, incertezze=True
    , remove_q=True, use_lower_limits=False):
126     """
127     Funzione per il preprocessing del dataset.
128 
129     Questa funzione carica un dataset da un file CSV, effettua operazioni di
     pulizia e filtraggio,
130     e applica il metodo Monte Carlo per gestire le incertezze nelle
    magnitudini, se richiesto.
131 
132     Parametri:
133     - file_path (str): Percorso del file CSV contenente il dataset.
134     - dataset_name (str): Nome del dataset, che verra' aggiunto come colonna
     nel DataFrame.
135     - sigma (float): Valore di sigma utilizzato nella generazione dei
    campioni Monte Carlo
136       per la distribuzione troncata.
137     - n_samples (int): Numero di campioni Monte Carlo da generare per ogni
    riga del dataset.
138       Se n_samples = 0, il metodo Monte Carlo non viene applicato e gli
    errori sui colori vengono
139       calcolati direttamente tramite propagazione dell'errore.
140     - incertezze (bool, default=True): Se True, applica il metodo Monte
    Carlo per gestire
141       le incertezze nei dati.
142     - remove_q (bool, default=True): Se True, rimuove dal dataset gli
    oggetti con Target "Q"
143       (quasar).
144     - use_lower_limits (bool, default=False): Se True, utilizza i valori
    limite inferiori per le
145       magnitudini censurate invece di stimarle con i colori medi.
146 
147     Ritorna:
148     - pd.DataFrame: DataFrame preprocessato con le eventuali trasformazioni
    applicate.
149     """
```

```
150
151     # Carica il dataset da file CSV
152     df = pd.read_csv(file_path)
153
154     # Aggiunge una colonna 'Dataset' con il nome del dataset per tenere
    traccia della provenienza dei dati
155     df['Dataset'] = dataset_name
156
157     # Rimuove gli oggetti con Target "Q" se specificato
158     if remove_q:
159         df = df[df['Target'] != 'Q']
160
161     # Rimuove gli oggetti con Target "OIII", indipendentemente dall'opzione
    remove_q
162     df = df[df['Target'] != 'OIII']
163
164     # Applica il metodo Monte Carlo per gestire le incertezze, se richiesto
165     if incertezze:
166         df = generate_samples(df, sigma, n_samples=n_samples,
    use_lower_limits=use_lower_limits)
167
168         # Se n_samples e' 0, calcola direttamente gli errori sui colori con
    la propagazione dell'errore
169         if n_samples == 0:
170             df['i-z_err'] = np.sqrt(df['i_err']**2 + df['z_err']**2)
171             df['z-y_err'] = np.sqrt(df['z_err']**2 + df['y_err']**2)
172             df['i-y_err'] = np.sqrt(df['i_err']**2 + df['y_err']**2)
173
174     # Calcola i colori i-z, z-y e i-y per ogni riga
175     df['i-z'] = df['i'] - df['z']
176     df['z-y'] = df['z'] - df['y']
177     df['i-y'] = df['i'] - df['y']
178
179     return df
180
181
182 def random_grid(X_train, y_train):
183     """
184     Funzione per eseguire la ricerca randomizzata degli iperparametri per un
     modello RandomForest.
185
186     Questa funzione utilizza `RandomizedSearchCV` per esplorare diverse
    combinazioni di iperparametri
187     e trovare la configurazione ottimale per un modello
    RandomForestClassifier.
```

```
188
189     Parametri:
190     - X_train (pd.DataFrame o np.array): Matrice delle feature di training.
191     - y_train (pd.Series o np.array): Array con le etichette di training.
192
193     Ritorna:
194     - random_search (RandomizedSearchCV): Oggetto `RandomizedSearchCV` con
    il miglior modello trovato.
195     """
196
197     # Definire lo spazio degli iperparametri da esplorare
198     param_grid = {
199         'n_estimators': np.arange(10, 110, 10),  # Numero di alberi nella
    foresta (tra 10 e 100 con passo 10)
200         'max_depth': np.arange(3, 16, 1),       # Profondita' massima degli
    alberi (da 3 a 15)
201         'min_samples_split': [5, 10],           # Minimo numero di campioni
    richiesti per dividere un nodo interno
202         'min_samples_leaf': [2, 4],             # Minimo numero di campioni
    necessari in una foglia terminale
203     }
204
205     # Creazione del classificatore RandomForest di base (senza iperparametri
     predefiniti)
206     rf = RandomForestClassifier()
207
208     # Configurare la ricerca randomizzata degli iperparametri
209     random_search = RandomizedSearchCV(
210         estimator=rf,               # Modello da ottimizzare
211         param_distributions=param_grid,  # Spazio degli iperparametri da
    esplorare
212         n_iter=100,                 # Numero totale di combinazioni casuali
    da testare
213         cv=3,                        # Numero di fold per la cross-
    validation (3-fold CV)
214         verbose=2,                   # Livello di dettaglio della stampa
    dell'output
215         n_jobs=-1                    # Usa tutti i processori disponibili
    per velocizzare la ricerca
216     )
217
218     # Addestra il modello su X_train e y_train esplorando le combinazioni di
     iperparametri
219     random_search.fit(X_train, y_train)
220
```

```
221     # Restituisce l'oggetto RandomizedSearchCV con i migliori parametri
        trovati
222     return random_search
223
224 def evaluate_with_monte_carlo(model, X_test, y_test, names_test, features,
        path, val_or_test, n_samples):
225     """
226     Valuta un modello di classificazione utilizzando il metodo Monte Carlo
        per gestire le incertezze.
227
228     Questa funzione esegue la classificazione su un dataset di test in cui
        ogni oggetto e' stato
229     campionato 'n_samples' volte tramite il metodo Monte Carlo. La classe
        finale assegnata a ogni
230     oggetto e' determinata dalla maggioranza delle predizioni fatte sui
        campioni generati.
231
232     Parametri:
233     - model (sklearn classifier): Modello addestrato da valutare.
234     - X_test (pd.DataFrame): Feature del dataset di test.
235     - y_test (pd.Series): Etichette reali del dataset di test.
236     - names_test (pd.Series): Colonna che identifica gli oggetti unici nel
        test set (nome o ID).
237     - features (list): Lista delle feature da usare per la classificazione.
238     - path (str): Percorso della cartella in cui salvare i risultati (
        confusion matrix e classification report).
239     - val_or_test (str): Stringa che specifica se il test set e' un
        validation set o un test set ("val" o "test").
240     - n_samples (int): Numero di campioni Monte Carlo generati per ogni
        oggetto nel test set.
241
242     Ritorna:
243     - accuracy (float): Accuratezza del modello calcolata sulle predizioni
        finali per ogni oggetto.
244     """
245
246     # Ottiene le etichette di classe uniche nel dataset di test
247     labels = np.unique(y_test)
248
249     # Liste per memorizzare le predizioni finali e le etichette originali
        degli oggetti
250     y_pred_mc = []
251     y_test_mc = []
252
253     # Itera su ogni oggetto unico nel dataset di test
```

```
    for name in names_test.unique():
        # Trova gli indici delle righe corrispondenti a questo oggetto in X_test e y_test
        object_indices = names_test[names_test == name].index

        # Effettua la predizione del modello per i 'n_samples' campioni di questo oggetto
        mc_predictions = model.predict(X_test.loc[object_indices, features])

        # Trova la classe piu' frequente tra le predizioni usando Counter
        majority_class = Counter(mc_predictions).most_common(1)[0][0]

        # Memorizza la classe assegnata con la regola della maggioranza
        y_pred_mc.append(majority_class)

        # Memorizza la classe reale dell'oggetto (tutte le righe hanno la stessa etichetta)
        original_class = y_test.loc[object_indices].values[0]
        y_test_mc.append(original_class)

    # Calcola l'accuratezza confrontando le predizioni maggioritarie con i target reali
    accuracy = accuracy_score(y_test_mc, y_pred_mc)

    # Genera la confusion matrix
    cm = confusion_matrix(y_test_mc, y_pred_mc, labels=labels)

    # Visualizza la confusion matrix come heatmap
    plt.figure()
    ax = plt.subplot()
    conf_matrix = sn.heatmap(cm, annot=True, ax=ax, fmt="d", cmap="Blues")

    # Etichette degli assi
    ax.set_xlabel("Predicted")
    ax.set_ylabel("True")
    ax.set_title("Confusion Matrix")
    ax.xaxis.set_ticklabels(labels)
    ax.yaxis.set_ticklabels(labels)

    # Salva la confusion matrix come immagine
    plt.savefig(f"{path}/confusion_matrix_media_{val_or_test}.png")
    plt.clf()  # Pulisce la figura per evitare sovrapposizioni nelle future visualizzazioni

    # Genera il classification report
```

```
    report = classification_report(y_test_mc, y_pred_mc, target_names=labels, output_dict=True)

    # Converte il classification report in un DataFrame e lo salva come CSV
    report_df = pd.DataFrame(report).transpose()
    report_df.to_csv(f"{path}/classification_report_media_{val_or_test}.csv")

    return accuracy

def apply_smote(X, y):
    """
    Applica l'algoritmo SMOTE (Synthetic Minority Over-sampling Technique) per bilanciare
    le classi nel dataset attraverso la generazione di nuovi campioni sintetici.

    SMOTE viene utilizzato per aumentare il numero di esempi della classe minoritaria
    creando nuovi campioni artificiali invece di duplicare i dati esistenti.

    Parametri:
    - X (pd.DataFrame o np.ndarray): Matrice delle feature, ovvero le variabili indipendenti del dataset.
    - y (pd.Series o np.ndarray): Vettore dei target, ovvero le etichette di classe.

    Ritorna:
    - X_res (pd.DataFrame o np.ndarray): Nuova matrice delle feature dopo l'applicazione di SMOTE.
    - y_res (pd.Series o np.ndarray): Nuovo vettore dei target con classi bilanciate.
    """

    # Inizializza SMOTE con un valore fisso per il random state per garantire la riproducibilita'
    smote = SMOTE(random_state=42)

    # Applica SMOTE per riequilibrare le classi generando nuovi campioni sintetici
    X_res, y_res = smote.fit_resample(X, y)

    # Restituisce il dataset riequilibrato
    return X_res, y_res

```

```
328 def random_grid_prf(X_train, dX_train, y_train, n_iter=10):
329     """
330     Esegue la ricerca casuale degli iperparametri per il modello PRF (
    Probabilistic Random Forest)
331     selezionando casualmente i valori da un set predefinito.
332 
333     A differenza di RandomizedSearchCV, questa funzione sceglie direttamente
     un set casuale
334     di iperparametri e addestra il modello con essi.
335 
336     Parametri:
337     - X_train (pd.DataFrame o np.ndarray): Matrice delle feature del
    training set.
338     - dX_train (pd.DataFrame o np.ndarray): Matrice delle incertezze
    associate alle feature.
339     - y_train (pd.Series o np.ndarray): Vettore dei target, ovvero le
    etichette di classe.
340 
341     Ritorna:
342     - model (PRF): Modello PRF addestrato con i parametri selezionati
    casualmente.
343     """
344 
345     # Definizione della griglia di ricerca degli iperparametri
346     param_grid = {
347         'n_estimators': np.arange(10, 110, 10),  # Numero di alberi nella
    foresta (da 10 a 100 con step di 10)
348         'max_depth': np.arange(3, 16, 1),        # Profondita' massima degli
     alberi (da 3 a 15 con step di 1)
349     }
350 
351     # Selezione casuale degli iperparametri dalla griglia definita
352     params = {
353         'n_estimators': np.random.choice(param_grid['n_estimators']),
354         'max_depth': np.random.choice(param_grid['max_depth'])
355     }
356 
357     # Inizializzazione del modello PRF con gli iperparametri selezionati
358     model = prf(**params)
359 
360     # Addestramento del modello sui dati di training con incertezze
361     model.fit(X_train, dX_train, y_train)
362 
363     # Restituisce il modello addestrato
364     return model
```

Listing C.3: Dettaglio dei parametri

```

# Definizione dei parametri di preprocessing per il training set
sigma_train = 5  # Numero di sigma per la distribuzione Monte Carlo sul
    training set
n_samples_train = 5  # Numero di campioni da generare per ogni oggetto nel
    training set
use_lower_limits = True  # Se considerare i limiti inferiori per le
    magnitudini censurate
incertezze = True  # Se integrare le incertezze nella generazione dei dati

# Definizione dei parametri di preprocessing per il test set
sigma_test = 2  # Numero di sigma per la distribuzione Monte Carlo sul test
    set
n_samples_test = 10  # Numero di campioni da generare per ogni oggetto nel
    test set
```

Esempio addestramento modello RF; RF + SMOTE; RF + LDA e RF + SMOTE + LDA, con applicazione di LLM per magnitudini non rilevavate

```

# Crea un DataFrame vuoto per memorizzare i risultati per ogni iterazione
iter_result_lda = pd.DataFrame(columns=["iter", "accuracy_test_lda", "
    accuracy_test_montecarlo_lda", "accuracy_val_lda", "
    accuracy_val_montecarlo_lda",
                                        "accuracy_test_smote_lda", "
    accuracy_test_montecarlo_smote_lda", "accuracy_val_smote_lda", "
    accuracy_val_montecarlo_smote_lda"])

# Inizia un ciclo che si ripete per 100 iterazioni
for i in range(100):
    print(i)  # Stampa il numero dell'iterazione corrente

    # Definisce il percorso dove salvare i risultati per questa iterazione
    path_base = 'LLM'

    # Crea la directory se non esiste gia'
    if not os.path.exists(path_base):
        os.mkdir(path_base)

    # Preprocessa il dataset SHELLQs applicando Monte Carlo con 5 campioni
```

```
    df_shellqs_MC = preprocessing("HSC.csv", "SHELLQs", sigma=5, incertezze=True, n_samples=5, use_lower_limits=True)

    # Preprocessa il dataset J1030 applicando Monte Carlo con 10 campioni
    df_j1030_MC = preprocessing("J1030.csv", "J1030", sigma=2, incertezze=True, n_samples=10, use_lower_limits=True)

    # Seleziona le features di interesse per i dataset SHELLQs e J1030
    features = ["i", "z", "y", "i-z", "z-y", "i-y"]

    # Estrae le features e i target per il dataset SHELLQs
    X_shellqs = df_shellqs_MC[features]
    y_shellqs = df_shellqs_MC['Target']
    names_shellqs = df_shellqs_MC['Name']  # Ottieni la colonna 'Name'

    # Estrae le features e i target per il dataset J1030 (test set)
    X_test = df_j1030_MC[features]
    y_test = df_j1030_MC['Target']
    names_test = df_j1030_MC['Name']  # Ottieni la colonna 'Name'

    # Crea un DataFrame con tutte le informazioni per SHELLQs
    data_shellqs = pd.DataFrame({
        'X': X_shellqs.to_dict('records'),  # Converte le righe di X_shellqs in dizionari
        'y': y_shellqs,
        'group': names_shellqs
    })

    # Raggruppa per 'group' e ottieni la classe predominante per ciascun gruppo
    grouped_shellqs = data_shellqs.groupby('group').agg({
        'y': lambda x: x.mode()[0],  # Classe predominante nel gruppo
        'X': 'count'  # Numero di campioni nel gruppo
    }).reset_index()

    # Esegui la stratificazione sui gruppi
    train_groups, val_groups = train_test_split(
        grouped_shellqs,
        test_size=0.2,
        stratify=grouped_shellqs['y'],  # Stratifica sulla classe predominante
        random_state=i
    )

    # Ricostruisci i dataset di training e validation filtrando i gruppi
```

```
    train_data = data_shellqs[data_shellqs['group'].isin(train_groups['group'])]
    val_data = data_shellqs[data_shellqs['group'].isin(val_groups['group'])]

    # Estrai X, y e nomi dai dati di training e validation
    X_train = pd.DataFrame(train_data['X'].tolist())  # Converte i dizionari di feature in DataFrame
    y_train = train_data['y'].reset_index(drop=True)
    names_train = train_data['group'].reset_index(drop=True)

    X_val = pd.DataFrame(val_data['X'].tolist())  # Converte i dizionari di feature in DataFrame
    y_val = val_data['y'].reset_index(drop=True)
    names_val = val_data['group'].reset_index(drop=True)

    # Risultati con LDA
    lda = LDA(n_components=1)  # Crea il modello LDA con 1 componente
    X_train_lda = lda.fit_transform(X_train, y_train)  # Applica LDA sui dati di training
    X_val_lda = lda.transform(X_val)  # Applica LDA sui dati di validazione
    X_test_lda = lda.transform(X_test)  # Applica LDA sui dati di test

    # Crea un DataFrame con una sola colonna per i dati trasformati da LDA
    X_train_lda_df = pd.DataFrame(X_train_lda, columns=['lda_component'])
    X_val_lda_df = pd.DataFrame(X_val_lda, columns=['lda_component'])
    X_test_lda_df = pd.DataFrame(X_test_lda, columns=['lda_component'])

    # Modello con LDA - addestriamo il Random Forest sui dati trasformati da LDA
    rf_random_lda = random_grid(X_train_lda_df, y_train)  # Esegui una ricerca random per il miglior modello
    best_random_lda = rf_random_lda.best_estimator_  # Ottieni il miglior modello

    # Percorso per salvare i risultati di questa iterazione
    path_LDA = f'{path_base}/LDA/result_{i}'
    if not os.path.exists(f'{path_base}/LDA'):
        os.mkdir(f'{path_base}/LDA')
    if not os.path.exists(path_LDA):
        os.mkdir(path_LDA)

    # Valutazione del modello LDA sui dati di test e validation
    accuracy_test_lda = evaluate(best_random_lda, X_test_lda_df, y_test, features=['lda_component'], path=path_LDA, val_or_test="test")
    accuracy_test_montecarlo_lda = evaluate_with_monte_carlo(best_random_lda
```

```
, X_test_lda_df, y_test, names_test, features=['lda_component'], path=
path_LDA, val_or_test="test", n_samples=10)
 accuracy_val_lda = evaluate(best_random_lda, X_val_lda_df, y_val,
features=['lda_component'], path=path_LDA, val_or_test="val")
 accuracy_val_montecarlo_lda = evaluate_with_monte_carlo(best_random_lda,
 X_val_lda_df, y_val, names_val, features=['lda_component'], path=
path_LDA, val_or_test="val", n_samples=5)

 # Risultati con SMOTE
 X_train_smote, y_train_smote = apply_smote(X_train_lda_df, y_train)  #
Applica SMOTE sui dati di training
 rf_random_smote_lda = random_grid(X_train_smote, y_train_smote)  #
Esegui una ricerca random per il miglior modello SMOTE
 best_random_smote_lda = rf_random_smote_lda.best_estimator_  # Ottieni
il miglior modello SMOTE

 # Percorso per salvare i risultati di questa iterazione con SMOTE
 path_LDA_SMOTE = f'{path_base}/LDA_SMOTE/result_{i}'
 if not os.path.exists(f'{path_base}/LDA_SMOTE'):
     os.mkdir(f'{path_base}/LDA_SMOTE')
 if not os.path.exists(path_LDA_SMOTE):
     os.mkdir(path_LDA_SMOTE)

 # Valutazione del modello SMOTE sui dati di test e validation
 accuracy_test_smote_lda = evaluate(best_random_smote_lda, X_test_lda_df,
 y_test, features=['lda_component'], path=path_LDA_SMOTE, val_or_test="
test")
 accuracy_test_montecarlo_smote_lda = evaluate_with_monte_carlo(
best_random_smote_lda, X_test_lda_df, y_test, names_test, features=['
lda_component'], path=path_LDA_SMOTE, val_or_test="test", n_samples=10)
 accuracy_val_smote_lda = evaluate(best_random_smote_lda, X_val_lda_df,
y_val, features=['lda_component'], path=path_LDA_SMOTE, val_or_test="val"
)
 accuracy_val_montecarlo_smote_lda = evaluate_with_monte_carlo(
best_random_smote_lda, X_val_lda_df, y_val, names_val, features=['
lda_component'], path=path_LDA_SMOTE, val_or_test="val", n_samples=5)

 # Aggiunge i risultati dell'iterazione al DataFrame
 iter_result_lda.loc[i] = [i, accuracy_test_lda,
accuracy_test_montecarlo_lda, accuracy_val_lda,
accuracy_val_montecarlo_lda,
                           accuracy_test_smote_lda,
accuracy_test_montecarlo_smote_lda, accuracy_val_smote_lda,
accuracy_val_montecarlo_smote_lda]

```

```
    # Salva il modello LDA per questa iterazione
    joblib.dump(lda, f'{path_LDA}/lda_model_{i}.pkl')

# Salva i risultati finali in un file CSV
iter_result_lda.to_csv("LLM/iter_result_lda.csv")
```

Esempio addestramento modello RF; RF + SMOTE; RF + LDA e RF + SMOTE + LDA, con applicazione di CBE per magnitudini non rilevavate

```

# Crea un DataFrame vuoto per memorizzare i risultati per ogni iterazione
iter_result_lda = pd.DataFrame(columns=["iter", "accuracy_test_lda", "
    accuracy_test_montecarlo_lda", "accuracy_val_lda", "
    accuracy_val_montecarlo_lda",
                                        "accuracy_test_smote_lda", "
    accuracy_test_montecarlo_smote_lda", "accuracy_val_smote_lda", "
    accuracy_val_montecarlo_smote_lda"])

# Inizia un ciclo che si ripete per 100 iterazioni
for i in range(100):
    print(i)  # Stampa il numero dell'iterazione corrente

    # Definisce il percorso dove salvare i risultati per questa iterazione
    path_base = 'CBE'
    os.makedirs(path_base, exist_ok=True)

    # Preprocessa i dataset
    df_shellqs_MC = preprocessing("HSC.csv", "SHELLQs", sigma=5, incertezze=
    True, n_samples=5)
    df_j1030_MC = preprocessing("J1030.csv", "J1030", sigma=2, incertezze=
    True, n_samples=10)

    # Seleziona le features
    features = ["i", "z", "y", "i-z", "z-y", "i-y"]

    # Estrai features e target
    X_shellqs, y_shellqs, names_shellqs = df_shellqs_MC[features],
    df_shellqs_MC['Target'], df_shellqs_MC['Name']
    X_test, y_test, names_test = df_j1030_MC[features], df_j1030_MC['Target'
    ], df_j1030_MC['Name']

```

```
    # Prepara il training e validation set
    train_data, val_data = stratified_group_split(X_shellqs, y_shellqs, names_shellqs, test_size=0.2, random_state=i)
    X_train, y_train, names_train = train_data
    X_val, y_val, names_val = val_data

    # Applica LDA
    lda = LDA(n_components=1)
    X_train_lda, X_val_lda, X_test_lda = lda.fit_transform(X_train, y_train), lda.transform(X_val), lda.transform(X_test)

    # Addestramento Random Forest con LDA
    best_random_lda = train_random_forest(X_train_lda, y_train)
    path_LDA = f'{path_base}/LDA/result_{i}'
    os.makedirs(path_LDA, exist_ok=True)

    # Valutazione
    accuracy_test_lda = evaluate(best_random_lda, X_test_lda, y_test, path=path_LDA, val_or_test="test")
    accuracy_test_montecarlo_lda = evaluate_with_monte_carlo(best_random_lda, X_test_lda, y_test, names_test, path=path_LDA, val_or_test="test", n_samples=10)
    accuracy_val_lda = evaluate(best_random_lda, X_val_lda, y_val, path=path_LDA, val_or_test="val")
    accuracy_val_montecarlo_lda = evaluate_with_monte_carlo(best_random_lda, X_val_lda, y_val, names_val, path=path_LDA, val_or_test="val", n_samples=5)

    # Applicazione di SMOTE e nuovo addestramento
    X_train_smote, y_train_smote = apply_smote(X_train_lda, y_train)
    best_random_smote_lda = train_random_forest(X_train_smote, y_train_smote)
    path_LDA_SMOTE = f'{path_base}/LDA_SMOTE/result_{i}'
    os.makedirs(path_LDA_SMOTE, exist_ok=True)

    # Valutazione SMOTE
    accuracy_test_smote_lda = evaluate(best_random_smote_lda, X_test_lda, y_test, path=path_LDA_SMOTE, val_or_test="test")
    accuracy_test_montecarlo_smote_lda = evaluate_with_monte_carlo(best_random_smote_lda, X_test_lda, y_test, names_test, path=path_LDA_SMOTE, val_or_test="test", n_samples=10)
    accuracy_val_smote_lda = evaluate(best_random_smote_lda, X_val_lda, y_val, path=path_LDA_SMOTE, val_or_test="val")
    accuracy_val_montecarlo_smote_lda = evaluate_with_monte_carlo(best_random_smote_lda, X_val_lda, y_val, names_val, path=path_LDA_SMOTE,
```

```
    val_or_test="val", n_samples=5)

    # Salvataggio risultati
    iter_result_lda.loc[i] = [i, accuracy_test_lda,
    accuracy_test_montecarlo_lda, accuracy_val_lda,
    accuracy_val_montecarlo_lda,
                              accuracy_test_smote_lda,
    accuracy_test_montecarlo_smote_lda, accuracy_val_smote_lda,
    accuracy_val_montecarlo_smote_lda]

    joblib.dump(lda, f'{path_LDA}/lda_model_{i}.pkl')

# Salva i risultati finali in un file CSV
iter_result_lda.to_csv("CBE/iter_result_lda.csv")
```

Esempio addestramento modello PRF; PRF + SMOTE, con applicazione di LLM per magnitudini non rilevavate

```

# Crea un DataFrame per memorizzare i risultati per ogni iterazione
iter_result_prf = pd.DataFrame(columns=["iter", "accuracy_test", "
    accuracy_val"])
iter_result_prf_smote = pd.DataFrame(columns=["iter", "accuracy_test", "
    accuracy_val"])

# Inizia un ciclo per 100 iterazioni
for i in range(100):
    print(f"Iteration {i}")

    # Definisce i percorsi per salvare i risultati per questa iterazione
    path_prf = f'result_prf_LLM/result_{i}'
    path_prf_smote = f'result_prf_smote_LLM/result_{i}'

    # Crea le directory se non esistono
    if not os.path.exists('result_prf_LLM'):
        os.makedirs('result_prf_LLM')
    if not os.path.exists(path_prf):
        os.makedirs(path_prf)

    if not os.path.exists('result_prf_smote_LLM'):
        os.makedirs('result_prf_smote_LLM')
    if not os.path.exists(path_prf_smote):
        os.makedirs(path_prf_smote)
```

```
24
25      # Preprocessa il dataset SHELLQs con Monte Carlo e 5 campioni
26      df_shellqs_MC = preprocessing("HSC.csv", "SHELLQs", sigma=5, n_samples
        =1, incertezze=True, remove_q=True, use_lower_limits=True)
27
28      # Preprocessa il dataset J1030 con Monte Carlo e 10 campioni
29      df_j1030_MC = preprocessing("J1030.csv", "J1030", sigma=2, n_samples=1,
        incertezze=True, remove_q=True, use_lower_limits=True)
30
31      # Definisce le features di interesse
32      features = ["i", "z", "y", "i-z", "z-y", "i-y"]
33      features_err = ["i_err", "z_err", "y_err", "i-z_err", "z-y_err", "i-
        y_err"]
34
35      # Divide il dataset SHELLQs in training e validation set
36      train_df, val_df = train_test_split(df_shellqs_MC, test_size=0.2,
        train_size=0.8, stratify=df_shellqs_MC['Target'])
37
38      # Estrae le features, incertezze e target per il dataset di training
39      X_train = train_df[features].reset_index(drop=True).to_numpy()
40      dX_train = train_df[features_err].reset_index(drop=True).to_numpy()
41      y_train = train_df['Target'].reset_index(drop=True).to_numpy()
42
43      # Applica SMOTE per bilanciare il dataset di addestramento
44      smote = SMOTE(random_state=42)
45      X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
46      dX_train_smote, _ = smote.fit_resample(dX_train, y_train)
47
48      # Estrae le features, incertezze e target per il dataset di validation
49      X_val = val_df[features].reset_index(drop=True).to_numpy()
50      dX_val = val_df[features_err].reset_index(drop=True).to_numpy()
51      y_val = val_df['Target'].reset_index(drop=True).to_numpy()
52
53      # Estrae le features e le incertezze per il dataset J1030
54      X_j1030 = df_j1030_MC[features].to_numpy()
55      dX_j1030 = df_j1030_MC[features_err].to_numpy()
56      y_j1030 = df_j1030_MC['Target'].to_numpy()
57
58      # Applica la ricerca manuale dei parametri per PRF
59      best_model = random_grid_prf(X_train, dX_train, y_train, n_iter=2)
60
61      # Valuta il modello sul validation set e salva l'accuratezza per il
        risultato senza SMOTE
62      accuracy_val = evaluate(best_model, X_val, dX_val, y_val, path=path_prf,
        val_or_test="val")
```

```
63
64      # Predice sui dati di J1030
65      accuracy_test = evaluate(best_model, X_j1030, dX_j1030, y_j1030, path=
        path_prf, val_or_test="test")
66
67      # Aggiunge i risultati dell'iterazione senza SMOTE al DataFrame
68      iter_result_prf.loc[i] = [i, accuracy_test, accuracy_val]
69
70      # Ora puoi usare X_train_smote e dX_train_smote per addestrare il
        modello
71      best_model_smote = random_grid_prf(X_train_smote, dX_train_smote,
        y_train_smote, n_iter=2)
72
73      # Valuta il modello sul validation set e salva l'accuratezza per il
        risultato con SMOTE
74      accuracy_val_smote = evaluate(best_model_smote, X_val, dX_val, y_val,
        path=path_prf_smote, val_or_test="val")
75
76      # Predice sui dati di J1030 con SMOTE
77      accuracy_test_smote = evaluate(best_model_smote, X_j1030, dX_j1030,
        y_j1030, path=path_prf_smote, val_or_test="test")
78
79      # Aggiunge i risultati dell'iterazione con SMOTE al DataFrame
80      iter_result_prf_smote.loc[i]=[i, accuracy_test_smote, accuracy_val_smote
        ]
81
82  # Salva i risultati finali in un file CSV per entrambi i casi
83  iter_result_prf.to_csv("result_prf_LLM/iter_result_prf.csv", index=False)
84  iter_result_prf_smote.to_csv("result_prf_smote_LLM/iter_result_prf_smote.csv
        ", index=False)
```

Esempio addestramento modello PRF; PRF + SMOTE, con applicazione di CBE per magnitudini non rilevavate

```
1
2   # Crea un DataFrame per memorizzare i risultati per ogni iterazione
3   iter_result_prf = pd.DataFrame(columns=["iter", "accuracy_test", "
        accuracy_val"])
4   iter_result_prf_smote = pd.DataFrame(columns=["iter", "accuracy_test", "
        accuracy_val"])
5
6   # Inizia un ciclo per 100 iterazioni
7   for i in range(100):
```

```
    print(f"Iteration {i}")

    # Definisce i percorsi per salvare i risultati per questa iterazione
    path_prf = f'result_prf_CBE/result_{i}'
    path_prf_smote = f'result_prf_smote_CBE/result_{i}'

    # Crea le directory se non esistono
    if not os.path.exists('result_prf_CBE'):
        os.makedirs('result_prf_CBE')
    if not os.path.exists(path_prf):
        os.makedirs(path_prf)

    if not os.path.exists('result_prf_smote_CBE'):
        os.makedirs('result_prf_smote_CBE')
    if not os.path.exists(path_prf_smote):
        os.makedirs(path_prf_smote)

    # Preprocessa il dataset SHELLQs con Monte Carlo e 5 campioni
    df_shellqs_MC = preprocessing("HSC.csv", "SHELLQs", sigma=5, n_samples
=1, incertezze=True, remove_q=True, use_lower_limits=False)

    # Preprocessa il dataset J1030 con Monte Carlo e 10 campioni
    df_j1030_MC = preprocessing("J1030.csv", "J1030", sigma=2, n_samples=1,
incertezze=True, remove_q=True, use_lower_limits=False)

    # Definisce le features di interesse
    features = ["i", "z", "y", "i-z", "z-y", "i-y"]
    features_err = ["i_err", "z_err", "y_err", "i-z_err", "z-y_err", "i-
y_err"]

    # Divide il dataset SHELLQs in training e validation set
    train_df, val_df = train_test_split(df_shellqs_MC, test_size=0.2,
train_size=0.8, stratify=df_shellqs_MC['Target'])

    # Estrae le features, incertezze e target per il dataset di training
    X_train = train_df[features].reset_index(drop=True).to_numpy()
    dX_train = train_df[features_err].reset_index(drop=True).to_numpy()
    y_train = train_df['Target'].reset_index(drop=True).to_numpy()

    # Applica SMOTE per bilanciare il dataset di addestramento
    smote = SMOTE(random_state=42)
    X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
    dX_train_smote, _ = smote.fit_resample(dX_train, y_train)

    # Estrae le features, incertezze e target per il dataset di validation
```

```
    X_val = val_df[features].reset_index(drop=True).to_numpy()
    dX_val = val_df[features_err].reset_index(drop=True).to_numpy()
    y_val = val_df['Target'].reset_index(drop=True).to_numpy()

    # Estrae le features e le incertezze per il dataset J1030
    X_j1030 = df_j1030_MC[features].to_numpy()
    dX_j1030 = df_j1030_MC[features_err].to_numpy()
    y_j1030 = df_j1030_MC['Target'].to_numpy()

    # Applica la ricerca manuale dei parametri per PRF
    best_model = random_grid_prf(X_train, dX_train, y_train, n_iter=2)

    # Valuta il modello sul validation set e salva l'accuratezza per il
    risultato senza SMOTE
    accuracy_val = evaluate(best_model, X_val, dX_val, y_val, path=path_prf,
    val_or_test="val")

    # Predice sui dati di J1030
    accuracy_test = evaluate(best_model, X_j1030, dX_j1030, y_j1030, path=
    path_prf, val_or_test="test")

    # Aggiunge i risultati dell'iterazione senza SMOTE al DataFrame
    iter_result_prf.loc[i] = [i, accuracy_test, accuracy_val]

    # Ora puoi usare X_train_smote e dX_train_smote per addestrare il
    modello
    best_model_smote = random_grid_prf(X_train_smote, dX_train_smote,
    y_train_smote, n_iter=2)

    # Valuta il modello sul validation set e salva l'accuratezza per il
    risultato con SMOTE
    accuracy_val_smote = evaluate(best_model_smote, X_val, dX_val, y_val,
    path=path_prf_smote, val_or_test="val")

    # Predice sui dati di J1030 con SMOTE
    accuracy_test_smote = evaluate(best_model_smote, X_j1030, dX_j1030,
    y_j1030, path=path_prf_smote, val_or_test="test")

    # Aggiunge i risultati dell'iterazione con SMOTE al DataFrame
    iter_result_prf_smote.loc[i]=[i, accuracy_test_smote, accuracy_val_smote
    ]

# Salva i risultati finali in un file CSV per entrambi i casi
iter_result_prf.to_csv("result_prf_CBE/iter_result_prf.csv", index=False)
iter_result_prf_smote.to_csv("result_prf_smote_CBE/iter_result_prf_smote.csv
```

```
    ", index=False)
```

Bibliografia

Balmaverde, B. et al. (2017). «The primordial environment of supermassive black holes (II): deep Y and J band images around the z=6.3 quasar SDSS J1030+0524». In: *Astronomy & Astrophysics* 606. DOI: 10.1051/0004-6361/201730683.

Bañados, E. et al. (2013). «THE GALAXY ENVIRONMENT OF A QSO AT $z \sim 5.7$». In: *The Astrophysical Journal* 773.2, p. 178. DOI: 10.1088/0004-637X/773/2/178.

Bañados, E. et al. (2018). «An 800-million-solar-mass black hole in a significantly neutral Universe at a redshift of 7.5». In: *Nature* 553, pp. 473–476. DOI: 10.1038/nature25180.

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. New York: Springer.

Bosman, S. E. I. et al. (2018). «New constraints on Lyman-α opacity with a sample of 62 quasars at z > 5.7». In: *Monthly Notices of the Royal Astronomical Society* 479, pp. 1055–1076. DOI: 10.1093/mnras/sty1652.

— (2022). «Hydr ogen r eionization ends by $z = 5.3$: Lymanα optical depth measur ed by the XQR-30 sample». In: *Monthly Notices of the Royal Astronomical Society* 514, pp. 55–75. DOI: https://doi.org/10.1093/mnras/stac1046.

Bosman, Sarah E. I. et al. (2020). «Three Lyα Emitting Galaxies within a Quasar Proximity Zone at z $\sim$ 5.8». In: *The Astrophysical Journal* 896.1, p. 49. DOI: 10.3847/1538-4357/ab85cd.

Bowler, R. A. A. et al. (2015). «The bright end of the galaxy luminosity function at $z \simeq 7$: before the onset of mass quenching?» In: *Monthly Notices of the Royal Astronomical Society* 440.3, pp. 2810–2842. DOI: 10.1093/mnras/stu449.

Breiman, Leo (2001). «Random Forests». In: *Machine Learning* 45.1, pp. 5–32. DOI: 10.1023/A:1010933404324.

Breiman, Leo et al. (1984). *Classification and Regression Trees*. Belmont, CA: Wadsworth.

Büyüköztürk, Ş. e Ö. Çokluk-Bökeoğlu (2008). «Discriminant Function Analysis: Concept and Application». In: *Egitim Arastirmalari - Eurasian Journal of Educational Research* 33, pp. 73–92.

Calzetti, D. et al. (2000). «The Dust Content and Opacity of Actively Star-forming Galaxies». In: *The Astrophysical Journal* 533.2, pp. 682–695. DOI: 10.1086/308692.

Champagne, J. B. et al. (2023). «A Mixture of LBG Overdensities in the Fields of Three $6 < z < 7$ Quasars: Implications for the Robustness of Photometric Selection». In: *Astronomy & Astrophysics* 952. DOI: 10.3847/1538-4357/acda8d.

Cohen, J. et al. (2003). *Applied Multiple Regression/Correlation Analysis for the Behavioural Sciences*. 3rd. Taylor & Francis Group.

Costa, Tiago, Debora Sijacki e Martin G. Haehnelt (2014). «Feedback from active galactic nuclei: energy- versus momentum-driving». In: *Monthly Notices of the Royal Astronomical Society* 444.3, pp. 2355–2376. DOI: 10.1093/mnras/stu1632.

Cutler, David R. et al. (2007). «Random forests for classification in ecology». In: *Ecological Applications* 17.6, pp. 1544–1556.

Decarli, R. et al. (2017). «Rapidly star-forming galaxies adjacent to quasars at redshifts exceeding 6». In: *Nature* 545, pp. 457–461. DOI: 10.1038/nature22358.

Di Matteo, Tiziana, Nishikanta Khandai et al. (2012). «Cold Flows and the First Quasars». In: *The Astrophysical Journal Letters* 745. DOI: 10.1088/2041-8205/745/2/L29.

Di Matteo, Tiziana, Volker Springel e Lars Hernquist (2005). «Energy input from quasars regulates the growth and activity of black holes and their host galaxies». In: *Nature* 433, pp. 604–607. DOI: 10.1038/nature03335.

Duda, Richard O., Peter E. Hart e David G. Stork (2001). *Pattern Classification*. Hoboken, NJ: Wiley-Interscience.

Dunlop, J. S. et al. (2013). «A deep ALMA image of the Hubble Ultra Deep Field». In: *Monthly Notices of the Royal Astronomical Society* 466.1, pp. 861–876. DOI: 10.1093/mnras/stw3088.

Eilers, A.-C., F. B. Davies e J. F. Hennawi (2018). «The Opacity of the Intergalactic Medium Measured along Quasar Sightlines at $z \sim 6$». In: *The Astrophysical Journal* 864, p. 53. DOI: 10.3847/1538-4357/aad4fd.

Fan, X. et al. (2006). «Constraining the Evolution of the Ionizing Background and the Epoch of Reionization with $z \sim 6$ Quasars. II. A Sample of 19 Quasars». In: *The Astronomical Journal* 132. DOI: 10.1086/504836.

Finkelstein, S. L. et al. (2015). «The Evolution of the Galaxy Rest-frame Ultraviolet Luminosity Function over the First Two Billion Years». In: *Astronomy & Astrophysics* 810. DOI: 10.1088/0004-637X/810/1/71.

Fisher, R. A. (1936). «The Use of Multiple Measurements in Taxonomic Problems». In: *Annals of Eugenics* 7.2, pp. 179–188. DOI: 10.1111/j.1469-1809.1936.tb02137.x.

Greig, B. e A. Mesinger (2017). «Simultaneously constraining the astrophysics of reionization and the epoch of heating with 21CMMC». In: *Monthly Notices of the Royal Astronomical Society* 472, pp. 2651–2669. DOI: 10.1093/mnras/stx2118.

Habouzit, Mélanie et al. (2019). «Supermassive black holes in cosmological simulations – II. The AGN luminosity and BH mass functions». In: *Monthly Notices of the Royal Astronomical Society* 489.1, pp. 1206–1220. DOI: 10.1093/mnras/stab3147.

Hastie, T., R. Tibshirani e J. Friedman (2009). *The Elements of Statistical Learning*. 2nd. Springer.

Hennawi, Joseph F. et al. (2006). «Binary Quasars in the Sloan Digital Sky Survey: Evidence for Excess Clustering on Small Scales». In: *The Astronomical Journal* 131, pp. 1–23. DOI: 10.1086/498235.

Ho, Tin Kam (1998). «Random decision forests». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.8, pp. 832–844.

Holtel, Frederik (feb. 2023). *Linear Discriminant Analysis (LDA) Can Be So Easy*. Retrieved: 2024-05-18. URL: https://medium.com/.

Huberty, Carl J. e Stephen Olejnik (2006). *Applied MANOVA and Discriminant Analysis*. Wiley-Interscience.

Izotov, Y. I., I. Orlitová et al. (2016). «Detection of high Lyman continuum leakage from four low-redshift compact star-forming galaxies». In: *Monthly Notices of the Royal Astronomical Society* 461.4, pp. 3683–3701. DOI: 10.1093/mnras/stw1205.

Izotov, Y. I., D. Schaerer et al. (2018). «Eight per cent leakage of Lyman continuum photons from a compact, star-forming dwarf galaxy». In: *Nature Astronomy* 2, pp. 374–377. DOI: 10.1038/nature16456.

J. A. Caballero, A. J. Burgasser e R. Klement (2008). «Contamination by field late-M, L, and T dwarfs in deep surveys». In: *A&A* 488.1. DOI: 10.1051/0004-6361:200809520.

Jolliffe, Ian T. (2002). *Principal Component Analysis*. New York: Springer.

Kim, S. et al. (2009). «The Environments of High-Redshift Quasi-Stellar Objects». In: *The Astrophysical Journal* 695.1, pp. 809–817. DOI: 10.1088/0004-637X/695/2/809.

Konno, Akira et al. (2014). «Accelerated Evolution of the Lyα Luminosity Function at $z > 7$ Revealed by the Subaru Ultra-Deep Survey for Lyα Emitters at $z = 7.3$». In: *The Astrophysical Journal* 797.1, p. 16. DOI: 10.1088/0004-637X/797/1/16.

Li, Yuexing et al. (2007). «Formation of z $\sim$ 6 Quasars from Hierarchical Galaxy Mergers». In: *The Astrophysical Journal* 665, pp. 187–208. DOI: 10.1086/519297.

Mardia, K. V., J. T. Kent e J. M. Bibby (1979). *Multivariate Analysis*. Academic Press.

Marley, Mark S. et al. (2021). «The Sonora Brown Dwarf Atmosphere and Evolution Models. I. Model Description and Application to Cloudless Atmospheres in Rainout Chemical Equilibrium». In: *The Astrophysical Journal* 920.2, p. 85. DOI: 10.3847/1538-4357/ac141d.

Mason, C. A. et al. (2018). «The Universe Is Reionizing at z $\sim$ 7: Bayesian Inference of the IGM Neutral Fraction Using Lyα Emission from Galaxies». In: *The Astrophysical Journal* 856. DOI: 10.3847/1538-4357/aab0a7.

Matsuoka, Y. et al. (2016). «Subaru High-z Exploration of Low-Luminosity Quasars (SHELLQs). I. Discovery of 15 Quasars and Bright Galaxies at 5.7 < z < 6.9». In: *The Astrophysical Journal* 828.1, p. 26. DOI: 10.3847/0004-637X/828/1/26.

— (2017). «Subaru High-z Exploration of Low-Luminosity Quasars (SHELLQs). II. Discovery of 32 Quasars and Luminous Galaxies at 5.7 < z ≤ 6.8». In: *The Astrophysical Journal* 837.2, p. L30. DOI: 10.1093/pasj/psx046.

— (2018). «Subaru High-z Exploration of Low-Luminosity Quasars (SHELLQs). IV. Discovery of 41 Quasars and Luminous Galaxies at 5.7 ≤ z ≤ 6.9». In: *The Astrophysical Journal Supplement Series* 237.1, p. 5. DOI: 10.3847/1538-4365/aac724.

— (2020). «Subaru High-z Exploration of Low-Luminosity Quasars (SHELLQs). X. Discovery of 35 Quasars and Luminous Galaxies at 5.7 ≤ z ≤ 7.0». In: *The Astrophysical Journal* 883.2, p. 183. DOI: 10.3847/1538-4357/ab3c60.

— (2022). «Subaru High-z Exploration of Low-Luminosity Quasars (SHELLQs). XVI. 69 New Quasars at 5.8 < z < 7.0». In: *The Astrophysical Journal Supplement Series* 259.1, p. 18. DOI: 10.3847/1538-4365/ac3d31.

Matthee, J. et al. (2022). «Re-solving reionization with Lyα: how bright Lyα emitters account for the z ≈2–8 cosmic ionizing background». In: *Monthly Notices of the Royal Astronomical Society* 512.4, pp. 5960–5977. DOI: 10.1093/mnras/stac801.

McGreer, I. D., A. Mesinger e V. D'Odorico (2015). «Model-independent evidence in favor of an end to reionization by z ≈6». In: *Monthly Notices of the Royal Astronomical Society* 447. DOI: 10.1093/mnras/stu2449.

McGreer, Ian D., Andrei Mesinger e Valentina D'Odorico (2016). «Model-independent evidence in favor of an end to reionization by z ≈6». In: *Monthly Notices of the Royal Astronomical Society* 455, pp. 72–77. DOI: 10.1093/mnras/stu2449.

McLachlan, G. J. (2004). *Discriminant Analysis and Statistical Pattern Recognition*. Wiley Interscience. ISBN: 978-0-471-69115-0.

Meyer, R. A. et al. (2022). «Physical Constraints on the Extended Interstellar Medium of the $z = 6.42$ Quasar J1148+5251: $[CII]_{158\mu m}$, $[NII]_{205\mu m}$, and $[OI]_{146\mu m}$ Observations». In: *The Astrophysical Journal* 927.2, p. 152. DOI: 10.3847/1538-4357/ac4e94.

Mignoli, M. et al. (2020). «Web of the giant: Spectroscopic confirmation of a large-scale structure around the $z = 6.31$ quasar SDSSJ1030+0524». In: *Astronomy & Astrophysics* 642, p. L1. DOI: 10.1051/0004-6361/202039045.

Morselli, L. et al. (2014). «Primordial environment of super massive black holes: large-scale galaxy overdensities around $z \sim 6$ quasars with LBT». In: *Astronomy & Astrophysics* 568, A1. DOI: 10.1051/0004-6361/201423853.

Mortlock, D. J. et al. (2011). «A luminous quasar at a redshift of z = 7.085». In: *Nature* 474, pp. 616–619. DOI: 10.1038/nature10159.

Murphy, Kevin P. (2012). *Machine Learning: A Probabilistic Perspective*. Cambridge, MA: MIT Press.

Nakajima, K. et al. (2020). «The VANDELS survey: the ionizing properties of star-forming galaxies at $3 \leq z \leq 5$ using deep rest-frame ultraviolet spectroscopy». In: *Monthly Notices of the Royal Astronomical Society* 493.3, pp. 5022–5044. DOI: 10.1093/mnras/stad1283.

Overzier, Roderik A. (2022). «Conditions for Direct Black Hole Seed Collapse near a Radio-loud Quasar 1 Gyr after the Big Bang». In: *The Astrophysical Journal* 926. DOI: 10.3847/1538-4357/ac448c.

Quinlan, J. R. (1986). «Induction of decision trees». In: *Machine Learning*. DOI: 10.1007/BF00116251.

Reis, Itamar, Dalya Baron e Sahar Shahaf (2018). «Probabilistic Random Forest: A Machine Learning Algorithm for Noisy Data Sets». In: *The Astronomical Journal* 157.1, p. 16. DOI: 10.3847/1538-3881/aaf101. URL: https://iopscience.iop.org/article/10.3847/1538-3881/aaf101/meta.

Robertson, B. E. et al. (2015). «Cosmic Reionization and Early Star-forming Galaxies: A Joint Analysis of New Constraints from Planck and the Hubble Space Telescope». In: *The Astrophysical Journal Letters* 802.2, p. L19. DOI: 10.1088/2041-8205/802/2/L19.

Russell, Stuart J. e Peter Norvig (2010). *Artificial Intelligence: A Modern Approach*. 3rd. Upper Saddle River, NJ: Prentice Hall.

Schenker, M. A. et al. (2012). «Keck Spectroscopy of Faint $3 < z < 8$ Lyman Break Galaxies: Evidence for a Declining Fraction of Emission Line Sources in the Redshift Range $6 < z < 8$». In: *The Astrophysical Journal* 744.2, p. 179. DOI: 10.1088/0004-637X/744/2/179.

Shen, Yue et al. (2007). «Clustering of High-Redshift ($z \geq 2.9$) Quasars from the Sloan Digital Sky Survey». In: *The Astronomical Journal* 133, pp. 2222–2241. DOI: 10.1086/513517.

Simpson, C. et al. (2014). «No excess of bright galaxies around the redshift 7.1 quasar ULAS J1120+0641». In: *Monthly Notices of the Royal Astronomical Society* 442.4, pp. 3454–3462. DOI: 10.1093/mnras/stu1116.

Springel, Volker, Tiziana Di Matteo e Lars Hernquist (2005). «Modelling feedback from stars and black holes in galaxy mergers». In: *Monthly Notices of the Royal Astronomical Society* 361, pp. 776–794. DOI: 10.1111/j.1365-2966.2005.09238.x.

Stiavelli, M. et al. (2005). «Evidence of Primordial Clustering around the QSO SDSS J1030+0524 at $z = 6.28$». In: *The Astrophysical Journal* 622.1, pp. L1–L4. DOI: 10.1086/429406.

Tilvi, V. et al. (2014). «Rapid Decline of Lyα Emission toward the Reionization Era». In: *The Astrophysical Journal* 794.1, p. 5. DOI: 10.1088/0004-637X/794/1/5.

Vanzella, E. et al. (2009). «SPECTROSCOPIC OBSERVATIONS OF LYMAN BREAK GALAXIES AT REDSHIFTS ∼4, 5, AND 6 IN THE GOODS-SOUTH FIELD». In: *Monthly Notices of the Royal Astronomical Society* 695.2. DOI: 10.1088/0004-637X/695/2/1163.

Vanzella, E. et al. (2023). «Discovery of a Large-scale Lyα Nebula in a Protocluster at z = 3.3». In: *The Astrophysical Journal* 942, p. L17. DOI: 10.1093/mnras/stw2442.

Venemans, B. P. et al. (2019). «400 pc Imaging of a Massive Quasar Host Galaxy at a Redshift of 6.6». In: *The Astrophysical Journal* 874.2, p. L30. DOI: 10.3847/2041-8213/ab11cc.

Willott, C. J. et al. (2010). «The Canada-France High-z Quasar Survey: Nine New Quasars and the Luminosity Function at Redshift 6». In: *The Astronomical Journal* 139.3, pp. 906–918. DOI: 10.1088/0004-6256/139/3/906.

Willott, C. J. et al. (2005). «First Results from the Canada-France High-z Quasar Survey: Constraints on the $z = 6$ Quasar Luminosity Function and the Quasar Contribution to Reionization». In: *The Astrophysical Journal* 633. DOI: 10.1086/462408.

Yang, J. et al. (2020). «A Luminous Quasar at Redshift 7.642». In: *The Astrophysical Journal Letters* 897, p. L14. DOI: 10.3847/2041-8213/abd8c6.

Yue, Minghao et al. (2021). «A Giant Lyα Nebula Associated with a Galaxy Overdensity at z = 5.7». In: *The Astrophysical Journal* 911, p. 44. DOI: 10.3847/1538-4357/aa5d14.

Zheng, W. et al. (2006). «An Overdensity of Galaxies near the Most Distant Radio-loud Quasar». In: *The Astrophysical Journal* 640.2, pp. 574–581. DOI: 10.1086/500167.

Zhu, G. et al. (2022). «Long Dark Gaps in the Lyβ Forest at $z < 6$: Evidence of Ultra-late Reionization from XQR-30 Spectra». In: *The Astrophysical Journal* 932, p. 214. DOI: 10.3847/1538-4357/ac6e60.

Ringraziamenti

Voglio ringraziare il Prof. Cristian Vignali e i ricercatori Fabio Vito e Marco Mignoli per il loro supporto e la disponibilità durante il lavoro di tesi. I loro consigli e la loro guida sono stati fondamentali per la realizzazione di questo progetto.