UNIVERSITÀ DEGLI STUDI DI PALERMO

DIPARTIMENTO DI INGEGNERIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

AI-DRIVEN LOAN AND MORTGAGE EVALUATION

TESI DI LAUREA DI
FRANCESCO DI LIBERTI

RELATORE
Prof. MARCO LA CASCIA

CONTRORELATORE
Prof. ALESSANDRA DE PAOLA

ANNO ACCADEMICO 2024 – 2025

Indice

Introduzione

La concessione di prestiti e mutui è una delle attività principali svolte dal sistema bancario e finanziario. Tramite queste concessioni, gli istituti di credito sostengono economicamente i richiedenti. Tuttavia, ogni decisione di erogazione porta con sé un elemento di incertezza: la possibilità che il debitore non riesca a restituire quanto ricevuto. Questo rischio di insolvenza, se non valutato e gestito in modo accurato, può generare gravi conseguenze per il singolo istituto, ma anche per la stabilità dell'intero sistema finanziario.

La stima del rischio di credito è una pratica che permette a un istituto di calcolare la probabilità che un cliente non rispetti gli impegni di restituzione. Si tratta di un compito complesso che coinvolge diversi fattori: dalle condizioni anagrafiche e socioeconomiche del richiedente, alla sua storia creditizia, fino ad aspetti più ampi legati al contesto economico e lavorativo. L'obiettivo è individuare, con il massimo grado di affidabilità possibile, quali soggetti possano rappresentare un rischio elevato e quali, invece, offrano sufficienti garanzie di rimborso. Una valutazione corretta consente di ridurre le perdite per l'istituto erogante e di garantire condizioni eque e sostenibili ai clienti meritevoli di fiducia.

Negli ultimi decenni, le tecniche a disposizione degli istituti finanziari hanno subito un processo evolutivo costante. I modelli tradizionali hanno progressivamente lasciato spazio ad approcci statistici e matematici più sofisticati, in grado di trattare un numero crescente di variabili e di cogliere relazioni complesse tra i dati. Al contempo, la digitalizzazione e la diffusione di nuovi strumenti di pagamento hanno reso disponibili enormi quantità di informazioni che, nonostante rappresentino una risorsa preziosa, pongono nuove sfide in termini di interpretazione e gestione responsabile.

La sempre maggiore disponibilità di dati ha reso, quindi, necessario ripensare le modalità di analisi. L'obiettivo è quello di garantire che le previsioni, oltre che accurate, siano comprensibili e verificabili. La fiducia dei clienti e la conformità ai requisiti normativi richiedono che le decisioni prese non siano percepite come arbitrarie o dettate da meccanismi opachi. In un settore delicato come quello del credito, la possibilità di spiegare in modo chiaro le ragioni di un rifiuto o di un'approvazione rappresenta un requisito essenziale per la tutela del cliente e, soprattutto, per la credibilità dell'istituto.

Da ciò nasce la necessità di conciliare l'accuratezza delle valutazioni con la loro

spiegabilità. Un modello estremamente preciso ma poco interpretabile, rischia di minare la fiducia degli utenti e di non rispettare le normative; viceversa, un approccio semplice e facilmente comprensibile può risultare inadeguato di fronte alla complessità dei dati e dei fenomeni reali. Il vero progresso risiede dunque nella capacità di integrare prestazioni predittive elevate con un livello adeguato di trasparenza.

In questo contesto si inserisce il presente lavoro di tesi, svolto presso l'azienda *DATA Reply*. L'obiettivo è applicare strumenti tecnologici avanzati alla valutazione del rischio di credito, esplorando nuove modalità per combinare precisione e interpretabilità. L'idea alla base è quella di sperimentare un approccio che consenta di migliorare l'efficienza del processo decisionale e, al tempo stesso, di offrire motivazioni comprensibili delle scelte operate.

Il lavoro si propone, quindi, di esplorare la fattibilità di un modello sperimentale in grado di affrontare queste sfide, basato su una combinazione di tecniche di analisi dati e di strumenti di supporto alla decisione. L'attenzione è, quindi, rivolta al risultato predittivo e al processo con cui tale risultato viene raggiunto, con l'intento di costruire un sistema capace di adattarsi ai dati a disposizione, di apprendere da essi e di motivare in maniera trasparente le proprie valutazioni.

Gli aspetti innovativi della tesi possono essere sintetizzati in tre punti fondamentali: la ricerca di un equilibrio tra accuratezza predittiva e trasparenza delle decisioni; la sperimentazione su dati complessi reali di un approccio che integra modelli quantitativi e strumenti di spiegazione delle scelte; la realizzazione di un prototipo dimostrativo che consente di esplorare e comprendere il funzionamento del sistema proposto.

Il documento è organizzato come segue. Nel **Capitolo 1** viene presentato lo stato dell'arte e il quadro teorico di riferimento, con un'analisi delle metodologie tradizionali e delle più recenti evoluzioni nel campo della valutazione del rischio di credito. Il **Capitolo 2** descrive il caso di studio e i dati impiegati, soffermandosi sulle caratteristiche del dataset e sulle principali sfide che esso comporta. Nel **Capitolo 3** viene illustrata la progettazione della pipeline proposta per la valutazione del credito, evidenziandone le componenti principali e la logica complessiva. Il **Capitolo 4** approfondisce gli aspetti implementativi e lo sviluppo della soluzione, con particolare attenzione alle scelte tecniche adottate. Nel **Capitolo 5** sono riportati i risultati sperimentali ottenuti e la relativa discussione critica, finalizzata a valutarne l'efficacia e i possibili miglioramenti futuri.

Capitolo 1

Stato dell'Arte e Fondamenti Teorici

1.1 Il problema del credit scoring

La valutazione del rischio di credito rappresenta una delle sfide più delicate nell'ambito dell'offerta di prestiti e mutui da parte di istituzioni finanziarie e bancarie. Con il termine *credito* si intende generalmente una somma di denaro concessa a un consumatore da un ente finanziatore, da restituire secondo un piano rateale, con scadenze prestabilite e con l'aggiunta di interessi. Dunque, il *credit scoring* può definirsi come il meccanismo cardine mediante il quale gli istituti stimano la probabilità che il richiedente sia in grado di adempiere agli obblighi di rimborso entro i tempi previsti [14]. Questo processo decisionale, che guida l'istituto nella scelta di accettare o meno la domanda di credito, si fonda sulle informazioni fornite dal richiedente e su quelle raccolte tramite diverse fonti (tra cui registri pubblici e database privati).

Il tema del credit scoring ha assunto, negli ultimi anni, una rilevanza crescente non solo per ragioni di efficienza operativa e controllo del rischio, ma anche a seguito delle recenti evoluzioni normative europee, che pongono requisiti stringenti in termini di trasparenza e responsabilità delle decisioni automatizzate. In particolare con il nuovo quadro regolatorio introdotto dal Regolamento (UE) 2024/1689 sull'Intelligenza Artificiale[1.1] [21] e le linee guida dell'European Banking Authority (EBA) in materia di concessione e monitoraggio dei prestiti [12]. Queste ultime sono state recepite e attuate dalla Banca d'Italia attraverso la Nota n. 13 del 20 luglio 2021 [4].

L'obiettivo principale di un sistema di credit scoring è, quindi, quello di attribuire un punteggio che rifletta la probabilità di inadempienza (*probability of default*) per ogni soggetto richiedente, sulla base di informazioni quantitative e qualitative, raccolte

[1.1] Il Regolamento (UE) 2024/1689, noto come AI Act, è il primo quadro normativo europeo sull'intelligenza artificiale. Stabilisce obblighi specifici per l'uso di sistemi di AI ad alto rischio, tra cui quelli impiegati per l'assegnazione del credito.

dall'istituto di credito sia da istituti pubblici, in Italia dalla Centrale dei Rischi [1.2], che da enti privati a cui l'istituto può fare riferimento per avere maggiori informazioni sul richiedente. La valutazione effettuata utilizzando la metodologia prescelta dall'istituto, sulla base delle politiche interne di decisione e della regolamentazione sopra citata, fornisce un risultato che contribuisce in modo diretto alla mitigazione del rischio di credito e all'efficienza e competitività dell'istituto bancario.

1.2 Metodi tradizionali di valutazione del rischio

Storicamente, le prime soluzioni per il credit scoring si sono basate su metodi statistici e modelli lineari. Tra questi si annoverano la regressione logistica, l'analisi discriminante lineare (LDA) e, successivamente, modelli ad alberi decisionali. Questi strumenti hanno permesso alle banche di passare da valutazioni soggettive, affidate all'esperienza del responsabile dell'istruttoria creditizia, a processi più standardizzati e replicabili [14]. Esempi di questi meccanismi replicabili sono le scorecard e le regole decisionali:

- Le scorecard costituiscono la metodologia tradizionale per la valutazione automatica del rischio di credito: si basano sull'assegnazione di un punteggio a ciascun richiedente, calcolato come somma pesata di variabili categoriche e numeriche (es. età, reddito, storico dei rapporti, ecc.), suddivise in classi e ciascuna associata a un contributo predefinito. Lo score totale viene poi confrontato con una soglia, che determina l'esito della valutazione (approvazione, rifiuto, richiesta di ulteriori garanzie). Questo approccio, basato su regressione logistica e metodi statistici, ha il pregio di essere trasparente e facilmente spiegabile, in linea con quanto richiesto dalle linee guida regolamentari [25].

- Le regole decisionali consistono in insiemi di condizioni logiche prefissate (ad esempio: "se il reddito è inferiore a una determinata soglia e l'anzianità lavorativa risulta bassa, allora rifiutare la domanda"). Tali regole, tipicamente definite sulla base di policy interne e dell'esperienza storica degli operatori, costituiscono un meccanismo semplice e interpretabile per automatizzare la decisione. Tuttavia, questi sistemi soffrono di limiti in termini di flessibilità e capacità di adattamento rispetto alle moderne tecniche data-driven. Le strategie più avanzate di estrazione automatica di regole interpretabili e il loro impiego nel credit scoring saranno approfonditi nella Sezione 1.5, dedicata all'algoritmo BRCG.

La diffusione di sistemi informativi più avanzati e la tendenza sempre maggiore verso la digitalizzazione del settore hanno favorito l'introduzione progressiva di tecniche

[1.2] La Centrale dei Rischi è un sistema informativo pubblico gestito dalla Banca d'Italia, che raccoglie informazioni sui rapporti di credito concessi dalle banche e dagli intermediari finanziari ai propri clienti. Maggiori dettagli e normativa di riferimento disponibili su: `https://www.bancaditalia.it/statistiche/raccolta-dati/centrale-rischi`.

più sofisticate, tra cui support vector machine (SVM), reti neurali e, in alcuni casi, algoritmi evolutivi e genetici [9]. Tuttavia, le scorecard restano ancora oggi uno standard industriale in molti contesti, poichè, come già evidenziato, si tratta di elementi che garantiscono trasparenza e facilità di spiegazione ai fini regolamentari, come previsto dalle linee guide EBA [12].

1.3 Machine Learning e AI nel settore finanziario

Con l'aumento del volume e dell'eterogeneità dei dati disponibili (anagrafici, transazionali, comportamentali e relazionali), il problema del credit scoring è progressivamente divenuto un campo privilegiato per le tecniche di Machine Learning e Intelligenza Artificiale, in quanto consentono di modellare relazioni non lineari complesse e sfruttare interazioni tra variabili, offrendo miglioramenti significativi in termini di capacità predittiva rispetto ai modelli tradizionali [18].

L'utilizzo di tecniche di Machine Learning nasce dall'esigenza di processare grandi moli di dati, evidenziando pattern ed eventuali segnali di rischio che non sarebbero facilmente individuabili da metodi lineari. Inoltre, grazie alla loro capacità di adattarsi a nuovi dati e di integrare informazioni da fonti differenti, i modelli di ML si sono resi strumenti fondamentali nella valutazione del rischio.

Le soluzioni ML più avanzate, oggi utilizzate a livello internazionale nei principali istituti finanziari, includono modelli di ensemble come Gradient Boosting Machines (ad esempio LightGBM [17] e XGBoost [8])[1.3], Random Forest [6] e, più recentemente, Deep Neural Networks applicate a dati strutturati e non strutturati[1.4] [26].

Questi approcci consentono, inoltre, un'importante riduzione dei tempi di risposta e dei costi operativi, permettendo una gestione più efficiente dei processi di valutazione e monitoraggio del rischio. È importante sottolineare, tuttavia, che l'introduzione massiva di modelli di ML pone nuove sfide soprattutto in termini di interpretabilità e trasparenza delle decisioni, aspetti che verranno approfonditi nelle sezioni successive.

Nel progetto qui presentato, il sistema impiega un modello supervisionato LightGBM [1] per la stima della probabilità di default.

1.4 Explainable AI e modelli interpretabili

L'evoluzione normativa europea e la crescente attenzione di regolatori, clienti e stakeholder verso la trasparenza delle decisioni automatizzate hanno reso imprescindibile

[1.3] Le Gradient Boosting Machines sono modelli di apprendimento ensemble che combinano deboli predittori (tipicamente alberi decisionali) per ottenere un modello forte. Maggiori informazioni disponibili su: `https://explained.ai/gradient-boosting/index.html`.

[1.4] Le reti neurali profonde (DNN) possono essere applicate a dati non strutturati come testo, immagini e audio, rendendole adatte all'analisi di documenti bancari, estratti conto e comunicazioni dei clienti.

l'adozione di modelli interpretabili e di strumenti di *Explainable AI* (XAI) nel settore bancario e finanziario. Come già discusso alla sezione 1.1, le normative europee impongono requisiti stringenti in termini di trasparenza, giustificabilità e controllo umano sulle decisioni algoritmiche, soprattutto in processi ad alto impatto sociale come la concessione del credito.

L'uso crescente di modelli predittivi complessi – i cosiddetti *black box models*[1.5], come molte soluzioni di Machine Learning e Deep Learning - solleva la problematica della comprensibilità delle decisioni: questi algoritmi possono fornire predizioni estremamente accurate, ma risultano spesso opachi e difficilmente interpretabili sia dagli operatori sia dagli stessi sviluppatori. Tale opacità rappresenta un limite significativo in ambito bancario, dove la possibilità di comprendere e giustificare le scelte algoritmiche è essenziale non solo per la conformità normativa, ma anche per la tutela dei clienti e la prevenzione di bias o discriminazioni sistematiche.

Per rispondere a queste esigenze, sono state sviluppate numerose tecniche e framework di explainability, in grado di attribuire un significato alle predizioni dei modelli complessi e di chiarire l'impatto delle singole variabili decisionali. Queste tecniche si suddividono generalmente in due categorie principali: approcci *explanation by design*, ovvero modelli in cui il contributo di ciascuna variabile alla predizione è direttamente leggibile e spiegabile [16] e metodi *post-hoc*, applicabili anche a modelli complessi già addestrati.

Tra questi ultimi, ampio utilizzo ha trovato LIME (*Local Interpretable Model-agnostic Explanations*), una tecnica che genera modelli locali interpretabili per spiegare individualmente ciascuna predizione di un modello complesso [23] e SHAP (*SHapley Additive exPlanations*), che sarà trattato in dettaglio nella Sezione 1.6. Questi strumenti permettono di fornire spiegazioni verificabili delle decisioni prese dai sistemi automatizzati, anche quando basati su architetture complesse.

L'utilizzo di queste soluzioni permette di rafforzare la fiducia nei sistemi AI applicati al credito, migliorando la trasparenza dei processi decisionali.

1.5 Algoritmi di Rule Mining: BRCG

Gli algoritmi di Rule Mining rappresentano una classe di tecniche che consentono di estrarre automaticamente regole decisionali interpretabili, fornendo spiegazioni chiare e facilmente comprensibili delle scelte compiute dai modelli predittivi.
Tra questi, l'algoritmo Boolean Decision Rules via Column Generation [10, 2] (BRCG), offre un approccio particolarmente efficace ed efficiente per l'identificazione di regole

[1.5] Con il termine *black box model* si fa riferimento a modelli predittivi il cui funzionamento interno non è immediatamente interpretabile, come reti neurali profonde o ensemble complessi. Una discussione sul tema è disponibile su: `https://www.ai4business.it/intelligenza-artificiale/explanaible-ai-comprendere-la-black-box/`.

decisionali booleane, basate su combinazioni di variabili binarie che descrivono il fenomeno analizzato.

Il modello BRCG è progettato specificamente per affrontare problemi di classificazione binaria, ovvero situazioni in cui l'obiettivo è assegnare ogni osservazione a una delle due possibili classi. Tale impostazione si adatta perfettamente al caso di studio affrontato in questa tesi, in cui la variabile target è rappresentata da un'etichetta binaria ottenuta a partire dalla probabilità stimata di default: al termine della fase di scoring, infatti, ciascun richiedente viene classificato come affidabile o a rischio sulla base di una soglia fissata sul punteggio di probabilità. La struttura binaria del problema, dunque, consente di sfruttare appieno le potenzialità di BRCG per la generazione di regole decisionali interpretabili, direttamente applicabili ai profili dei clienti oggetto di valutazione.

La metodologia BRCG è formulata come un problema di ottimizzazione combinatoria, affrontato mediante una tecnica nota come *column generation*[1.6]. Tale approccio consente di gestire in modo efficiente grandi insiemi di regole candidate, esplorando soltanto un sottoinsieme delle regole più promettenti in termini di accuratezza e semplicità. L'algoritmo permette di generare regole interpretabili espresse, a seconda delle esigenze, sia in *Disjunctive Normal Form* (DNF)[1.7] , ovvero come una disgiunzione di congiunzioni (regole del tipo "se almeno una di queste condizioni è verificata..."), sia in *Conjunctive Normal Form* (CNF)[1.8] , ossia come una congiunzione di disgiunzioni (regole del tipo "se tutte queste condizioni, ciascuna delle quali ammette diverse alternative, sono soddisfatte..."). Questa flessibilità di rappresentazione facilita l'adozione delle regole sia in sistemi automatici sia come strumento esplicativo a supporto delle decisioni, rendendole accessibili anche a operatori non tecnici.

Internamente, il modello BRCG implementa un meccanismo decisionale basato sulle regole estratte, il cui processo predittivo può essere interpretato attraverso l'explainer fornito dal modello stesso. Tale meccanisco intrinseco del modello costituisce un chiaro esempio di explainability by design, poiché le regole decisionali sono esplicitamente accessibili e comprensibili sin dalla fase di costruzione del modello, senza la necessità di strumenti di spiegazione post-hoc. Questo aspetto rende tale modello idoneo per applicazioni critiche in ambito finanziario, dove la possibilità di comprendere e giustificare in modo chiaro le decisioni algoritmiche è cruciale per il rispetto delle normative vigenti e per mantenere la fiducia degli stakeholder coinvolti.

[1.6] La *column generation* è una tecnica di programmazione matematica utilizzata per risolvere problemi di grandi dimensioni, costruendo iterativamente il modello con un sottoinsieme di variabili (colonne) rilevanti. Si veda: `https://optimization.cbe.cornell.edu/index.php?title=Column_generation_algorithms`.

[1.7] La forma normale disgiuntiva (DNF) è una rappresentazione logica in cui una formula è espressa come disgiunzione (OR) di congiunzioni (AND) di letterali.

[1.8] La forma normale congiuntiva (CNF) è una rappresentazione logica in cui una formula è espressa come congiunzione (AND) di disgiunzioni (OR) di letterali.

L'adozione di BRCG in questo lavoro è stata determinata proprio da tali proprietà di interpretabilità: le regole estratte sono facilmente integrabili con il sistema basato su Large Language Models (LLM), permettendo la generazione di predizioni trasparenti e motivate.

1.6 Tecniche di explainability: SHAP

Le tecniche di explainability post-hoc si sono affermate come strumenti indispensabili per la comprensione e la verifica delle decisioni prodotte da modelli complessi di machine learning. Tra queste, una delle più rilevanti e adottate in ambito accademico e industriale è SHAP [19] (*SHapley Additive exPlanations*). SHAP consente di attribuire in modo rigoroso e trasparente l'importanza di ciascuna variabile alle predizioni effettuate dal modello, sia a livello locale (ossia per la singola istanza) sia a livello globale (per l'intero dataset).

Alla base di SHAP vi è il concetto dei valori di Shapley[1.9], sviluppati nella teoria dei giochi cooperativi, che permette di valutare quale sia il contributo marginale di ogni variabile rispetto al risultato finale, considerando tutte le possibili combinazioni delle altre variabili in gioco. In pratica, SHAP quantifica come cambia la predizione del modello quando una certa caratteristica viene aggiunta o rimossa, garantendo così spiegazioni coerenti e teoricamente fondate.

Nel caso specifico del credit scoring, l'interpretazione dei valori SHAP ha un importanza decisiva: un valore SHAP positivo associato a una variabile spinge la predizione verso la classe positiva ("buon creditore"), mentre un valore SHAP negativo indica un contributo verso la classe negativa ("cattivo creditore"). Questo aspetto permette agli operatori, anche non tecnici, di comprendere quali fattori abbiano guidato la decisione del modello in ciascun caso, fornendo una spiegazione puntuale della valutazione ottenuta dal cliente.

Un ulteriore punto di forza di SHAP risiede nella possibilità di aggregare i valori su tutto il dataset, consentendo di stilare classifiche delle variabili più influenti sul comportamento generale del modello. In ambito finanziario, questa funzionalità è di particolare importanza per motivare le scelte operative, individuare potenziali fonti di bias e garantire la conformità alle normative.

L'utilizzo di SHAP rappresenta un elemento molto importante per lo sviluppo responsabile di modelli di machine learning nelle applicazioni bancarie, rendendo possibile un controllo effettivo sulle decisioni automatizzate e contribuendo a consolidare la fiducia degli stakeholder nei confronti delle tecnologie di AI.

[1.9] I valori di Shapley sono un concetto fondamentale della teoria dei giochi cooperativi, introdotti da Lloyd Shapley nel 1953. Una spiegazione divulgativa è reperibile su: `https://www.rockettoride.com/xai/valori-shapley-e-la-teoria-dei-giochi-cooperativi`.

Capitolo 2

Descrizione del Caso di Studio e dei Dati

2.1 Contesto aziendale

Il progetto di tesi è stato svolto in collaborazione con **DATA Reply**[2.1], società appartenente al gruppo Reply e attiva a livello internazionale nella consulenza tecnologica e nei servizi di innovazione digitale, legati all'ambito di analisi e gestione dei dati. DATA Reply si articola in diverse Business Unit, ciascuna focalizzata su specifici settori di mercato e ambiti applicativi.

L'attività di ricerca è stata svolta all'interno della **Business Unit bancaria e finanziaria**[2.2], specializzata nello sviluppo di soluzioni data-driven e modelli di Intelligenza Artificiale avanzati per istituti di credito e intermediari finanziari. In questo contesto, il progetto di tesi si propone di contribuire all'ottimizzazione dei processi di valutazione del rischio creditizio, attraverso l'integrazione di modelli explainable AI e agenti conversazionali nel ciclo di approvazione di prestiti e mutui, in linea con le più recenti esigenze di trasparenza, robustezza e compliance richieste dal quadro normativo europeo (AI Act [21], linee guida EBA [12]).

2.2 Obiettivi e vincoli

Il progetto di tesi si inserisce nel contesto operativo del settore bancario e finanziario, dove la valutazione automatizzata dei profili dei richiedenti di prestiti e mutui rappresenta un nodo centrale per l'efficienza dei processi interni e la mitigazione del rischio di

[2.1] Maggiori informazioni sull'azienda: `https://www.reply.com/data-reply/it`.

[2.2] Una Business Unit è un'entità organizzativa autonoma all'interno di un'azienda, con responsabilità specifiche su una linea di prodotti, un settore di mercato o un'area applicativa. Nel caso di DATA Reply, ogni Business Unit è focalizzata su domini tecnologici o industriali differenti.

credito. L'obiettivo ultimo è consentire all'istituto di credito di operare in sicurezza, prendendo decisioni sostenibili, trasparenti e ben motivate.

In quest'ottica, il lavoro si propone di progettare e validare una pipeline AI-driven in grado di supportare il processo decisionale creditizio. Il sistema integra modelli di Machine Learning supervisionato con tecniche di explainability e una logica a due agenti basata su Large Language Models[2.3]. L'architettura, come verrà approfondito nel Capitolo 3, prevede un sistema ad agenti che si occuperanno di generare, applicare e validare regole, a supporto del processo di valutazione del rischio.

La soluzione progettata intende rispondere a una serie di obiettivi progettuali fondamentali:

- **Accuratezza e robustezza** delle predizioni, ottenute attraverso modelli supervisionati capaci di operare anche su dati imperfetti o incompleti, con un livello accettabile di generalizzazione;
- **Trasparenza e spiegabilità** delle decisioni, garantita sia a livello locale (per ogni singola predizione, mediante strumenti come SHAP, discussi nella sezione 1.6) sia a livello globale, grazie alla possibilità di motivare le valutazioni emesse dagli agenti;
- **Conformità normativa**, nel rispetto delle disposizioni europee e nazionali in materia di processo decisionale automatizzato, trasparenza e trattamento dei dati. In particolare, il sistema è progettato tenendo conto del Regolamento (UE) 2024/1689 sull'Intelligenza Artificiale (AI Act [21]), delle *Guidelines on loan origination and monitoring* dell'EBA [12], e delle *Disposizioni di Vigilanza* della Banca d'Italia [3] in tema di concessione del credito;
- **Adattabilità e automazione** nei flussi di validazione, grazie a un meccanismo di feedback tra predizione e conferma, che consente di aggiornare in modo dinamico e ragionato la knowledge base a partire dal confronto tra l'LLM e il modello di scoring.
- **Efficienza e scalabilità**, al fine di garantire l'applicabilità del sistema su dataset di grandi dimensioni e l' eventuale integrazione in contesti produttivi aziendali.

Accanto a questi obiettivi, il progetto ha tenuto conto di una serie di vincoli fondamentali:

- La necessità di preservare l'interpretabilità e la leggibilità dei modelli utilizzati e delle regole derivate, in modo da consentire un controllo umano costante e informato, anche da parte di personale non tecnico.

[2.3] I Large Language Models (LLM) sono modelli di deep learning addestrati su grandi quantità di testo, in grado di generare risposte linguisticamente coerenti e contestualmente rilevanti. Esempi noti sono GPT-4, Gemini e Claude.

- L'obbligo di mantenere coerenza e qualità nella knowledge base aggiornata dagli agenti, evitando derive completamente autonome che non siano verificabili a posteriori.
- L'attenzione ai principi di equità, imparzialità e non discriminazione nella valutazione dei richiedenti, in linea con le direttive europee sulla correttezza algoritmica e le buone pratiche di etica dell'utilizzo dell'intelligenza artificiale.

2.3 Descrizione del dataset

Il dataset utilizzato per la realizzazione di questo progetto proviene dalla competizione *Home Credit – Credit Risk Model Stability* organizzata su Kaggle [15]. Si tratta di una collezione di dati eterogenei, provenienti da fonti interne ed esterne, progettata per simulare uno scenario reale di valutazione del rischio di credito.

Il dataset è composto da numerose tabelle ad aggregazione multipla, suddivise su tre livelli di profondità informativa (`depth=0`, `depth=1`, `depth=2`). Le tabelle di livello 0 contengono dati statici associati a ogni richiesta di credito, mentre quelle di livello 1 e 2 forniscono informazioni storiche e relazionali (es. transazioni pregresse, informazioni su garanti o co-applicanti).

A partire dalla struttura multi-tabellare originaria, i dati in formato `parquet`[2.4] sono stati caricati e aggregati tramite una pipeline personalizzata, sviluppata a partire dal progetto di tesi di Matteo Altieri [1][2.5]

L'unione dei file, eseguita tramite opportune metodologie, è stata seguita da aggregazioni temporali e trasformazioni numeriche e categoriali dei predittori storici, con l'obiettivo di produrre un unico dataset strutturato e utilizzabile a valle per l'addestramento e la validazione del sistema predittivo.

Successivamente, i dati grezzi sono stati sottoposti a una fase strutturata di pulizia, trasformazione e selezione delle variabili, discusse nella sezione 3.2, al fine di migliorare la qualità informativa del dataset e assicurare robustezza nei passaggi successivi della pipeline.

Al termine della procedura, il dataset ottenuto è stato suddiviso in tre sottoinsiemi disgiunti:

- **Training set**, utilizzato per l'addestramento del modello scorecard e per la generazione delle regole tramite l'algoritmo BRCG, descritto in dettaglio nella sezione 1.5;

[2.4] Il formato `parquet` è un formato di archiviazione ottimizzato per sistemi distribuiti e carichi analitici, supportato da framework come Apache Spark . Maggiori dettagli su: `https://parquet.apache.org/`.

[2.5] Gli script di preprocessing, feature engineering e addestramento del modello LightGBM sono stati originariamente sviluppati da Matteo Altieri nel contesto della sua tesi di Laurea Magistrale. Nel presente lavoro, tali script sono stati adattati e modificati per rispondere alle specifiche esigenze progettuali e all'architettura descritta.

- **Validation set**, impiegato per la validazione della pipeline basata su agenti, in particolare per verificare la coerenza tra predizioni fornite dall'LLM e gli output prodotti dal modello di scoring, supportando l'aggiornamento dinamico delle regole nella knowledge base;
- **Test set**, riservato alla visualizzazione interattiva dei risultati e alla valutazione finale delle prestazioni del sistema su dati mai utilizzati in fase di addestramento o validazione.

Tutti i file intermedi sono stati esportati in formato `CSV` per facilitarne la portabilità e la riproducibilità.

2.4 Feature principali e loro significato

L'analisi dell'importanza delle variabili predittive, condotta tramite l'impiego degli indici SHAP, ha permesso di individuare le feature maggiormente determinanti nel processo di assegnazione del punteggio di rischio di default. In Figura 2.2 vengono mostrate, in ordine decrescente, le prime dieci variabili globali secondo l'importanza media assoluta dei valori SHAP.

Le variabili riportate nel grafico evidenziano la prevalenza di informazioni riconducibili a caratteristiche anagrafiche, lavorative e comportamentali del richiedente. In particolare, tra le feature più rilevanti si annoverano:

- **max_empl_employedfrom_271D**: Massima anzianità lavorativa rilevata tra tutte le fonti disponibili per il soggetto. Valori elevati di questa variabile riflettono una maggiore stabilità occupazionale, che generalmente è associata a un rischio creditizio inferiore.
- **max_employedfrom_700D**: Data di inizio del rapporto di lavoro più remota, recuperata dalle richieste di credito precedenti. Anche in questo caso, un'anzianità lavorativa maggiore è tipicamente indice di affidabilità.
- **meanAP_totalamount_6A_9**: Valore medio, calcolato su un determinato arco temporale e su più fonti, dell'importo totale dei contratti chiusi dal cliente. Una gestione regolare di volumi di credito rilevanti può indicare una maggiore esperienza finanziaria, ma anche un potenziale rischio in caso di sovraindebitamento.
- **last_birth_259D**: Data di nascita più recente tra tutti i soggetti collegati al contratto. Questa informazione, se riferita a cointestatari o garanti, può incidere sulla valutazione anagrafica e sulla prospettiva di rimborso a lungo termine.

- **mobilephncnt_593L**: Numero di soggetti collegati allo stesso numero di telefono mobile. Valori elevati possono suggerire situazioni di condivisione anomala del recapito, potenzialmente associate a tentativi di frode.

- **mean_numberofoverdueinstlmax_1151L_9**: Media del numero massimo di rate scadute su contratti chiusi, calcolata su diversi periodi di osservazione. Questa variabile riflette la storia dei ritardi nei pagamenti: valori alti sono fortemente correlati con una maggiore probabilità di insolvenza futura.

- **min_monthlyinstlamount_332A_9**: Importo minimo, su tutti i contratti attivi, della rata mensile sostenuta dal cliente. Può essere utilizzata per stimare la soglia minima di impegno finanziario già affrontato, utile per valutare la sostenibilità di nuove esposizioni.

- **numrejects9m_859L**: Numero di richieste di credito respinte nei nove mesi precedenti. La presenza di numerosi rifiuti da parte di altri istituti può essere un segnale precoce di criticità o di tentativi ripetuti di accesso al credito in condizioni di rischio.

- **avgdpdtolclosure24_3658938P**: Media dei giorni di ritardo (*Days Past Due*) sulle chiusure contrattuali degli ultimi 24 mesi. Una frequenza elevata di ritardi è un indicatore diretto di comportamento finanziario problematico e di rischiosità.

- **mean_numberofoverdueinstlmax_1039L_9**: Media del numero massimo di rate scadute su contratti attualmente attivi. Anche in questo caso, la presenza di rate scadute in essere rappresenta un fattore di rischio elevato.

La presenza di queste variabili tra le top-10 dimostra che il modello di scoring assegnato dal sistema recepisce segnali provenienti sia dal profilo storico e anagrafico sia dalle dinamiche comportamentali e creditizie più recenti. È importante sottolineare che le variabili riportate nel grafico rappresentano il risultato delle operazioni di pulizia, aggregazione e trasformazione dei dati descritte nella Sezione 3.2. Molte feature, infatti, derivano da aggregazioni statistiche (come massimo, media o minimo) calcolate su insiemi di informazioni storiche relative a ciascun cliente, come evidenziato dai prefissi `max_`, `mean_` e `min_`[2.6] . Inoltre, l'utilizzo degli indici SHAP consente non solo di identificare le feature più influenti, ma anche di fornire spiegazioni trasparenti e interpretabili a supporto delle decisioni automatizzate che saranno, in seguito, prese dagli agenti che aggiorneranno le regole per predire il profilo di rischio del cliente, come tratteremo nel Capitolo 3.

[2.6]Questi prefissi indicano il tipo di aggregazione statistica effettuata durante il preprocessing: `max_` per il valore massimo, `mean_` per la media, e `min_` per il valore minimo osservato nelle finestre temporali o sui set di relazioni storiche.

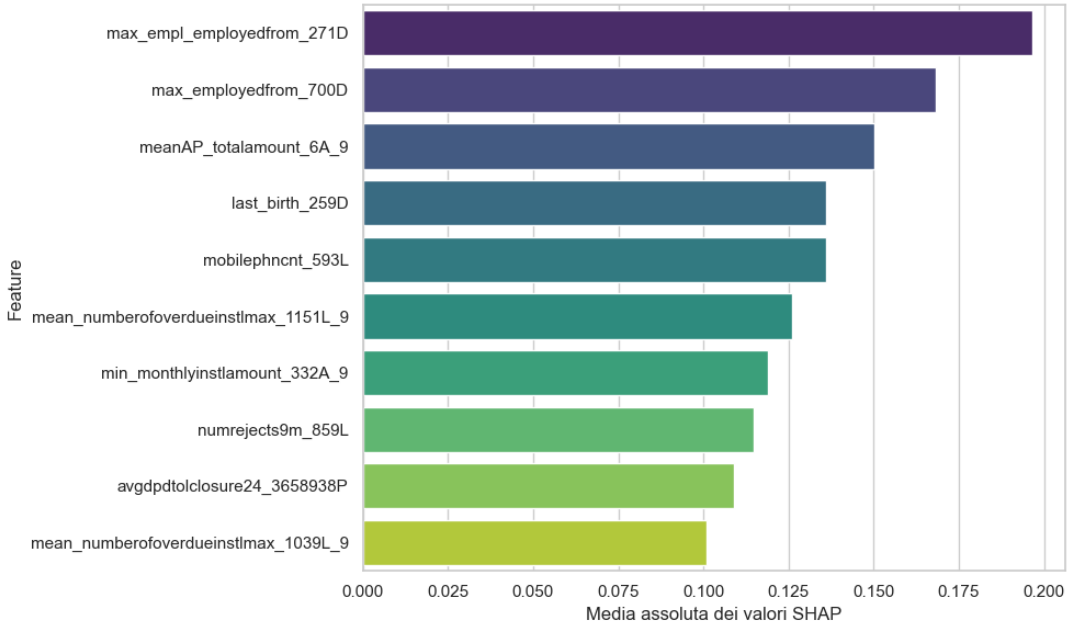

Figura 2.1: Importanza globale delle prime 10 variabili predittive secondo la media assoluta dei valori SHAP.

2.5 Analisi esplorativa preliminare

L'analisi esplorativa iniziale ha permesso di ottenere una visione d'insieme sulla struttura e sulla distribuzione delle principali variabili del dataset. Dopo la fase di pulizia, il dataset integrato risultante (completo, con i set di training, validazione e test) si compone di 1.232.695 osservazioni e 417 colonne, suddivise tra 285 variabili di tipo float64, 126 di tipo object e 6 di tipo int64.

Un elemento di particolare rilievo emerso dall'analisi riguarda la distribuzione del punteggio di rischio assegnato dal modello. Il `PD_score` rappresenta, per ciascun richiedente, la probabilità stimata di default[2.7], normalizzata su una scala continua tra 0 (assenza di rischio) e 1 (rischio massimo).

Come illustrato in Figura 2.2, la distribuzione del `PD_score` risulta asimmetrica:

- si osserva una densità elevata di soggetti con punteggio di rischio molto basso (valori prossimi allo zero), tipica di una popolazione generale in cui la maggior parte dei clienti soddisfa i criteri minimi di affidabilità creditizia;
- emerge un secondo picco attorno a valori intermedi (circa 0.35–0.45), riconducibile a richieste con caratteristiche particolari o a profili che presentano elementi di rischio moderato;

[2.7] La *Probabilità di Default*, rappresenta la probabilità che un cliente non sia in grado di adempiere agli obblighi di rimborso.

- la densità diminuisce progressivamente verso i valori più elevati di rischio (oltre 0.7), riflettendo la rarità di profili considerati ad alto rischio di default.

Questa distribuzione non uniforme riflette da un lato la selettività dei processi bancari reali, nei quali solo una piccola frazione delle richieste viene effettivamente giudicata come ad alto rischio; dall'altro lato, evidenzia la necessità di progettare modelli e metriche di valutazione capaci di operare in presenza di sbilanciamento naturale tra le classi[2.8]. In particolare, la predominanza di valori bassi e intermedi del `PD_score` suggerisce che il sistema di scoring attribuisce livelli di rischio crescenti solo in presenza di segnali robusti di fragilità finanziaria, coerentemente con le prassi prudenti adottate dagli istituti di credito.

Questa asimmetria costituisce un elemento centrale per tutto lo sviluppo del sistema, in quanto rende più complessa la discriminazione e l'apprendimento dei profili effettivamente ad alto rischio. In particolare, come sarà approfondito nel Capitolo 3 (con specifico riferimento alla Sezione 3.6.3 dedicata alla progettazione dei prompt e della logica agent-based), sarà fondamentale orientare il modello in modo da tollerare la presenza di alcuni falsi positivi-ossia classificare come "cattivo creditore" un soggetto in realtà affidabile-pur di ridurre al minimo i falsi negativi, che corrispondono all'approvazione di clienti ad alto rischio, in relazione agli obiettivi di tutela del rischio per l'istituto di credito.

2.6 Criteri per la scelta della soglia di classificazione

Un aspetto cruciale nella definizione della variabile target, denominata `rule_brcg`, per la valutazione del rischio di credito riguarda la determinazione della soglia per la binarizzazione applicata al punteggio predetto dal modello (`PD_score`), ovvero la probabilità stimata di default. Questo passaggio è necessario in quanto, mentre il ground truth del dataset originale fornisce etichette binarie (0 per "buon creditore" e 1 per "cattivo creditore"), il modello di scoring restituisce valori continui compresi tra 0 e 1. Di conseguenza, per tradurre tali punteggi in una classificazione discreta, è fondamentale individuare una soglia ottimale in grado di discriminare efficacemente tra le due classi, minimizzando il rischio di errore sia per i clienti affidabili sia per quelli ad alto rischio.

Per garantire la massima imparzialità nella selezione della soglia, è stata condotta un'analisi sistematica su un intervallo continuo di valori compresi tra 0,05 e 0,99. Per ciascuna soglia candidata, è stata calcolata la metrica macro-averaged F1-score

[2.8] Il problema di sbilanciamento delle classi (class imbalance) si verifica quando le classi target di un dataset non sono rappresentate in modo equo. È comune nei problemi di rischio di credito, dove i casi di default sono molto meno frequenti.

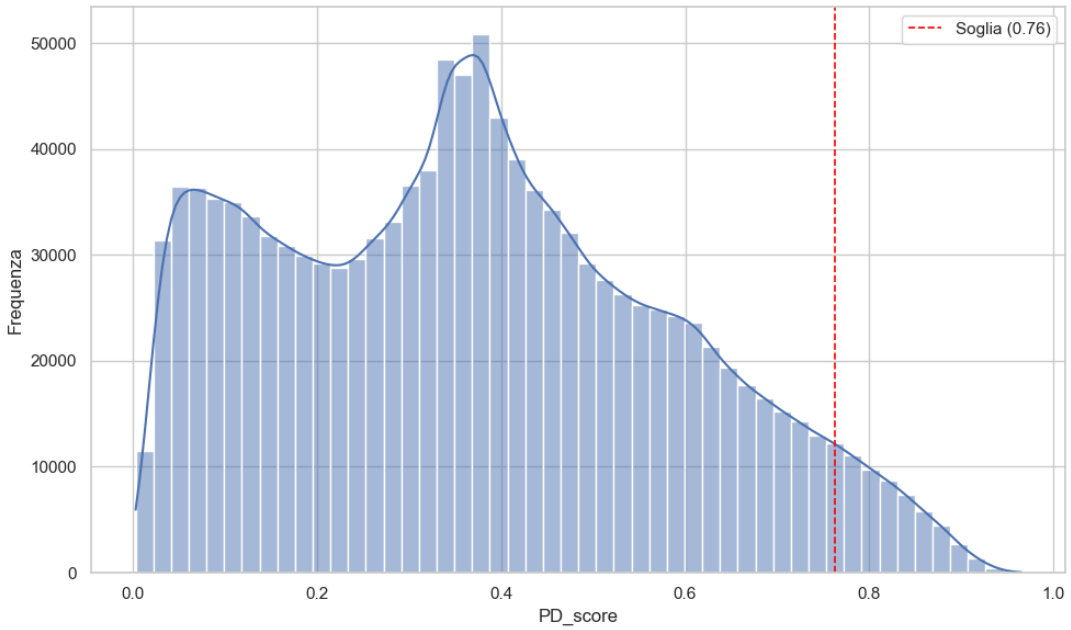

Figura 2.2: Distribuzione del punteggio di rischio PD nel dataset completo. Si evidenzia la presenza di due modalità principali: una concentrazione di punteggi bassi (clienti affidabili) e un picco su valori intermedi, con progressiva diminuzione dei casi a rischio elevato.

sull'intero dataset[2.9]. La soglia ottimale è stata identificata come quella in grado di massimizzare il macro F1-score; in caso di parità tra più candidati, è stato selezionato il valore che minimizza il numero totale di errori di classificazione (ovvero la somma di falsi positivi e falsi negativi).

La procedura ha restituito una soglia ottimale pari a 0,7621, arrotondata ai fini del progetto a 0,76, un valore sensibilmente più elevato rispetto alle soglie convenzionali (ad esempio 0.5 o 0.65), che riflette la natura sbilanciata del dataset, come si osserva dalla Figura 2.2, e le esigenze conservative proprie del contesto bancario.

Tutte le analisi successive, incluse la generazione delle regole e la validazione della pipeline agent-based, fanno riferimento a questa definizione della variabile target, assicurando coerenza e riproducibilità lungo tutte le fasi di sperimentazione e valutazione dei risultati.

[2.9]La scelta del macro F1-score come criterio di ottimizzazione è motivata dal fatto che rappresenta la media aritmetica degli F1-score calcolati separatamente per ciascuna classe, attribuendo quindi pari importanza sia alla classe maggioritaria che a quella minoritaria. Questo la rende particolarmente indicata nel nostro contesto caratterizzato da forte sbilanciamento, dove è essenziale garantire buone prestazioni anche sulla classe dei "cattivi creditori".

2.7 Problematiche specifiche: dati mancanti, sbilanciamento, qualità

L'analisi delle problematiche intrinseche al dataset ha evidenziato numerosi aspetti critici che hanno influenzato in modo significativo le scelte metodologiche adottate in fase di sviluppo. In primo luogo, la gestione di una mole di dati estremamente elevata (oltre 1.200.000 osservazioni e centinaia di variabili) ha reso necessario ricorrere a specifiche strategie di ottimizzazione della memoria per rendere efficiente la fase di preprocessing e preparare i dati all'addestramento del modello per il calcolo del `PD_score`. Tra queste strategie rientrano il downcasting[2.10] dei tipi numerici da 64 a 32, 16 o 8 bit e l'impiego della libreria `polars`[2.11], particolarmente adatta alla manipolazione di DataFrame di grandi dimensioni.

Un secondo aspetto di rilievo riguarda la presenza massiccia di valori mancanti: circa un centinaio delle colonne presenta oltre il 60% di dati assenti, mentre circa cinquanta variabili superano la soglia dell'80%. Di conseguenza, per preservare la qualità informativa del dataset ed evitare distorsioni introdotte da imputazioni non affidabili, si è preferito eliminare interamente le colonne maggiormente compromesse, limitando l'imputazione ai casi meno critici e secondo criteri di coerenza statistica.

Un ulteriore elemento di complessità è rappresentato dal forte sbilanciamento della variabile target. Come illustrato nella Figura 2.3, la suddivisione binaria del target — ottenuta applicando una soglia di 0,76 al punteggio di rischio `PD_score`, come sopra descritto — ha prodotto una netta prevalenza della classe 0 (clienti a basso rischio, "buoni creditori"), che rappresenta oltre 1.174.000 osservazioni, rispetto alla classe 1 (clienti ad alto rischio, "cattivi creditori"), che si attesta a circa 58.000 casi. In termini relativi, la classe minoritaria corrisponde dunque a circa $\frac{58{,}097}{1{,}174{,}598} \approx 4{,}9\%$ del totale delle osservazioni.

Come mostrato anche nella Figura 2.4, tale squilibrio si mantiene sostanzialmente invariato nei tre sottoinsiemi di training, validation e test, a garanzia della coerenza statistica fra gli split utilizzati nelle diverse fasi di sviluppo e valutazione.

Tale sbilanciamento ha reso necessario sviluppare specifiche metodologie implementative, sia per semplificare la generazione delle regole, sia per favorire un maggior bilanciamento nell'addestramento dei modelli e nella generazione degli explainer. Questi aspetti saranno approfonditi nel Capitolo 4.

[2.10] Il downcasting consiste nel ridurre il tipo di dato numerico per ridurre il consumo di memoria, a patto che la precisione sia compatibile con l'intervallo dei valori effettivi.

[2.11] `polars` è una libreria open source per la manipolazione di dati in DataFrame, scritta in Rust e progettata per alte prestazioni su dataset di grandi dimensioni. È un'alternativa più veloce rispetto a `pandas` in molti casi d'uso. Maggiori dettagli su: `https://www.pola.rs/`.

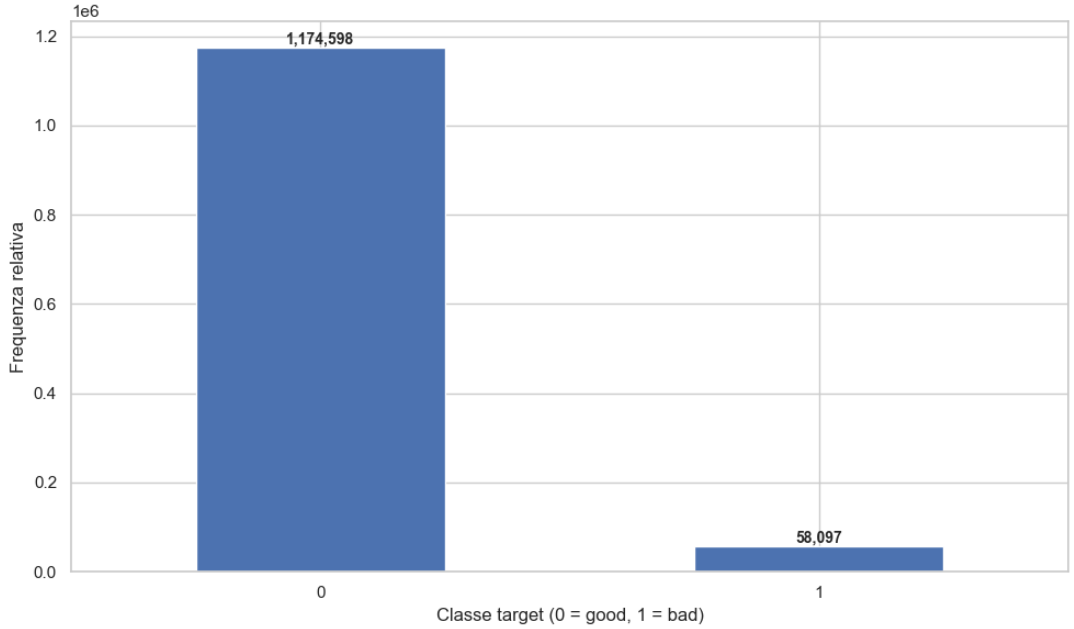

Figura 2.3: Distribuzione della classe target nel dataset: la classe 0 (buoni creditori) prevale nettamente rispetto alla classe 1 (cattivi creditori), riflettendo lo sbilanciamento tipico dei contesti reali di concessione del credito.

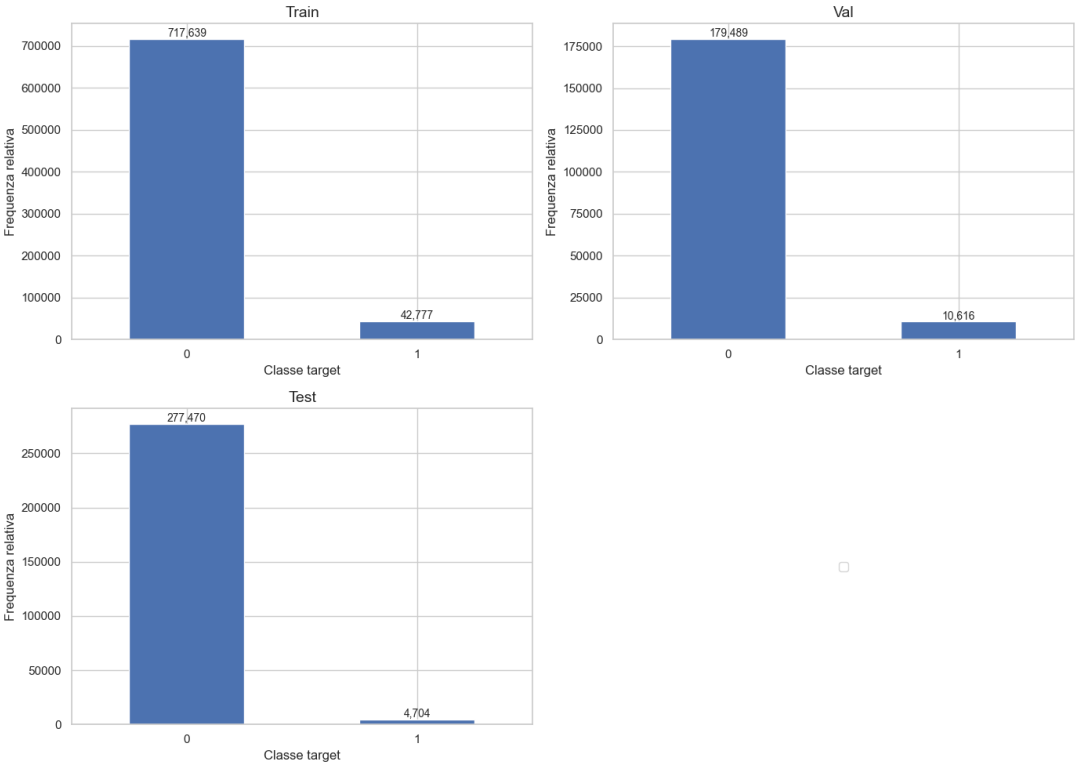

Figura 2.4: Distribuzione della classe target all'interno dei tre sottoinsiemi di training, validation e test: lo sbilanciamento tra buoni e cattivi creditori si mantiene costante in ciascuno split, riflettendo la distribuzione osservata sull'intero dataset.

Capitolo 3

Progettazione della Pipeline AI per la Valutazione del Credito

3.1 Schema architetturale generale

La struttura generale del progetto si sviluppa partendo dal caricamento dei dati bancari, forniti in formato parquet dalla competizione di riferimento [15]. Questi dati vengono sottoposti a una fase preliminare di preparazione, pulizia e feature engineering, approfondita all'interno della Sezione 3.2.

Al termine della prima fase di preparazione, il dataset viene utilizzato per addestrare un modello di machine learning basato su LightGBM. Contestualmente, viene generato un explainer SHAP, fondamentale per garantire la spiegabilità del sistema e l'individuazione delle feature più rilevanti nei processi decisionali.

Successivamente, i dati vengono ulteriormente elaborati e, infine, binarizzati, al fine di consentire la generazione di regole esplicite tramite l'algoritmo BRCG. Questo processo, articolato su più fold, produce un insieme di regole che, una volta aggregate, costituiscono la Knowledge Base (KB) iniziale utilizzata dagli agenti.

La parte centrale dell'architettura è rappresentata da una pipeline composta da due agenti LLM:

- Agente predittore: riceve in input il profilo del cliente e l'insieme di regole, restituendo una prima classificazione (buon/cattivo creditore) tramite ragionamento guidato, sulla base di punti di forza e debolezza;
- Agente validatore: confronta la predizione dell'agente LLM con il punteggio di rischio prodotto dal modello LightGBM, trasformato in classe binaria applicando il valore di soglia predefinito. In caso di discordanza, il validatore identifica le feature più influenti tramite SHAP e aggiorna dinamicamente la base di conoscenza, introducendo nuove regole mirate.

Questa architettura consente di integrare la flessibilità e trasparenza delle decisioni rule-based con la robustezza predittiva dei modelli, garantendo la tracciabilità delle decisioni e la possibilità di adattare in tempo reale le regole in risposta a nuovi casi.

L'intero flusso dati e la struttura delle componenti principali sono illustrati schematicamente nelle Figure 3.1 e 3.2, rispettivamente dedicate alla pipeline di Machine Learning e generazione delle regole e alla logica agent-based.

Tutti gli aspetti qui introdotti saranno approfonditi nelle sezioni successive del capitolo.

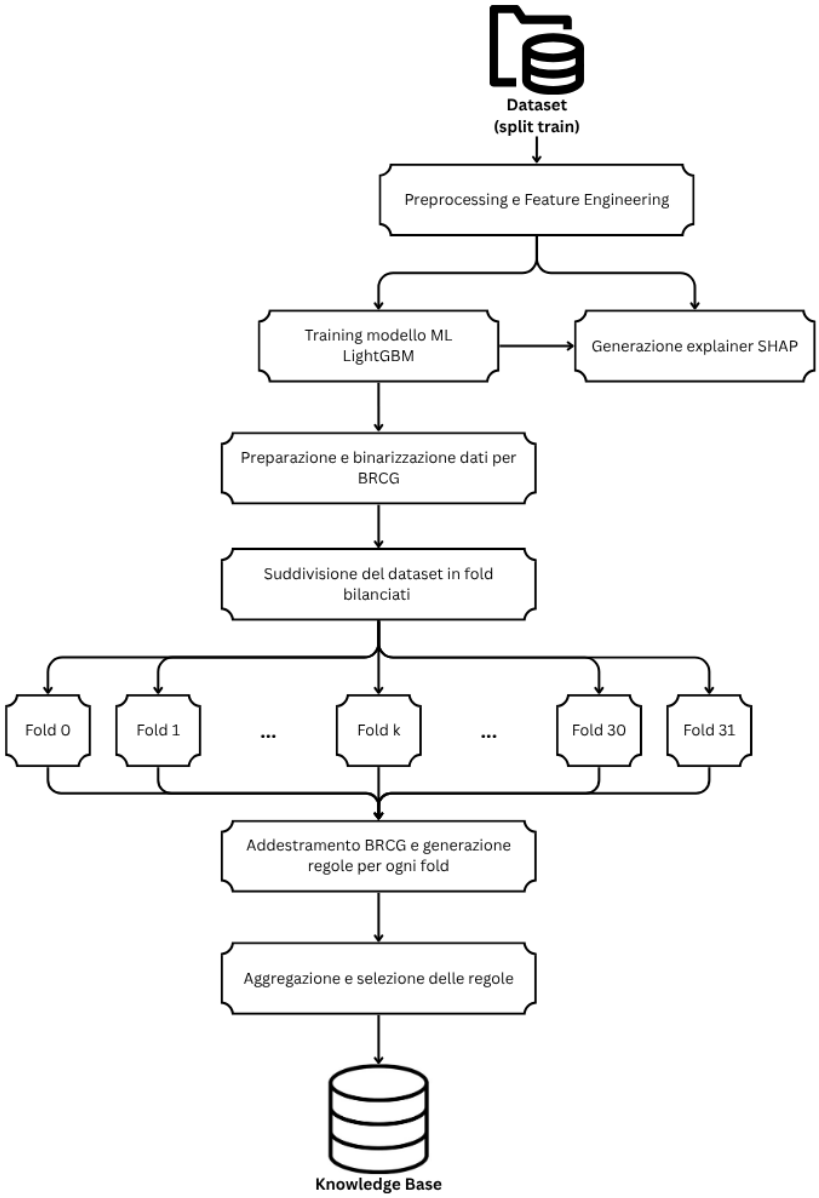

Figura 3.1: Schema della pipeline di preparazione dati, addestramento modello ML (LightGBM), generazione dell'Explainer SHAP e costruzione della Knowledge Base di regole tramite BRCG su fold multipli.

3.2 Preparazione dei dati e feature engineering

Le strategie di preprocessing e trasformazione dei dati adottate in questa sezione sono state ispirate e adattate da un precedente lavoro di tesi [1], con opportune modifiche motivate dal contesto applicativo e dalla specifica struttura dei dati utilizzati.

Le diverse fasi sono descritte nelle sezioni a seguire:

3.2.1 Caricamento dei dati

La fase iniziale del processo prevede il caricamento e la combinazione dei dati grezzi, che risultano suddivisi in molteplici file *parquet* provenienti da diverse fonti informative [15]. Il dataset è organizzato su tre diversi livelli di profondità informativa (come già brevemente discusso in sezione 2.3):

- **Depth 0:** dati statici e anagrafici di base relativi ai clienti e alle richieste di credito;
- **Depth 1:** informazioni aggiuntive relative a precedenti applicazioni, registri fiscali, dati provenienti da agenzie di credito, e altre fonti collegate direttamente al richiedente;
- **Depth 2:** ulteriori dati di dettaglio, come informazioni supplementari da credit bureau, applicazioni passate e dati personali ad un livello ancora più granulare.

I file vengono letti separatamente come dataframe Polars e successivamente integrati tramite operazioni di join, utilizzando come chiave primaria un identificativo univoco di ciascun caso (`case_id`). Questo approccio consente di ricostruire per ogni record un profilo informativo il più completo e ricco possibile, sfruttando appieno la profondità e la varietà delle fonti disponibili.

La procedura di caricamento assicura inoltre che ogni informazione sia processata e tipizzata in modo coerente rispetto alla sua natura (numerica, categorica, data, ecc.), facilitando così le successive fasi di trasformazione e pulizia. Infatti, l'obiettivo di questa fase è garantire che il dataset finale sia completo dal punto di vista informativo e strutturato secondo uno schema uniforme e facilmente gestibile nei passaggi successivi della pipeline.

3.2.2 Feature Engineering

La fase di *feature engineering* riveste un ruolo cruciale nella pipeline, in quanto consente di trasformare e arricchire i dati originari attraverso la creazione di nuove variabili, l'aggregazione di informazioni provenienti da fonti eterogenee e l'applicazione di regole di tipizzazione coerenti.

La procedura comprende alcune operazioni di **aggregazione** dei dati, sia a livello orizzontale (tra tabelle di diversa profondità) sia verticale (all'interno delle singole fonti), sfruttando funzioni di sintesi quali massimo, minimo, media e deviazione standard. In questo modo, si ottengono indicatori rappresentativi della storia creditizia e della posizione finanziaria del cliente, a partire da dati grezzi potenzialmente ridondanti o distribuiti su più fonti.

In questa fase è risultata fondamentale la definizione di **indici finanziari avanzati**, costruiti tramite opportune combinazioni tra variabili chiave (ad esempio, rapporto tra reddito e ammontare richiesto, rapporto di indebitamento, Loan-to-Value[3.1], Credit Utilization Rate[3.2], e altri indicatori di rischio creditizio). Tali indici, permettono di sintetizzare in modo efficace la solidità finanziaria e il livello di rischio associato a ciascun richiedente.

Particolare attenzione viene, inoltre, dedicata alla **gestione delle informazioni temporali**, con il calcolo di variabili derivate (come il mese e il giorno della settimana della decisione, o le differenze temporali tra eventi rilevanti) e la normalizzazione delle date rispetto ai principali momenti della pratica creditizia.

Infine, durante la costruzione delle nuove feature, vengono gestiti eventuali valori anomali (ad esempio divisioni per zero o infiniti) e garantita la coerenza dei dati per tutte le osservazioni del dataset. L'obiettivo complessivo di questa fase è quello di fornire, a valle della *feature engineering*, una rappresentazione del dato più ricca, strutturata e interpretabile, pronta per essere utilizzata nelle successive fasi di pulizia, trasformazione ed addestramento dei modelli.

3.2.3 Pulizia dei dati

Una volta completate le fasi di caricamento e di feature engineering, il dataset viene sottoposto a una rigorosa procedura di pulizia, finalizzata a garantire la qualità, la coerenza e l'efficienza dei dati che saranno utilizzati per l'addestramento dei modelli predittivi.

Il processo di data cleaning si articola nei seguenti principali passaggi:

- **Eliminazione delle variabili non informative:** vengono rimosse tutte le colonne che presentano una percentuale di valori mancanti superiore a una soglia prefissata (in questo progetto la soglia fissata è del 70%) oppure che risultano costanti (ovvero con un'unica modalità osservata), in quanto prive di contributo informativo per il modello;

[3.1] Il rapporto Loan-to-Value (LTV) rappresenta il rapporto tra l'ammontare del prestito richiesto e il valore del bene a garanzia (ad esempio un immobile). Valori elevati indicano un maggior rischio per il finanziatore.

[3.2] Il Credit Utilization Rate misura la percentuale di credito disponibile effettivamente utilizzata dal cliente. Un valore troppo elevato può indicare sovraindebitamento o uso intensivo del credito.

- **Conversione ed ottimizzazione della struttura dati:** i dati vengono convertiti da formato *Polars* a *Pandas* tramite una procedura a blocchi. In questa fase, particolare attenzione è dedicata alla corretta gestione delle variabili categoriche e binarie, che vengono tipizzate in modo esplicito, così da ottimizzare sia la memoria sia la velocità di elaborazione;

- **Riduzione dell'occupazione di memoria:** il dataset viene ulteriormente ottimizzato tramite il downcasting selettivo dei tipi numerici, adattando ogni variabile alla rappresentazione di minor dimensione compatibile con il suo dominio di valori;

- **Rimozione di periodi anomali:** al fine di garantire l'omogeneità temporale del dataset ed evitare bias legati a eventi eccezionali, vengono filtrate ed escluse tutte le osservazioni riferite a periodi successivi a marzo 2020, così da escludere effetti distorsivi dovuti alla pandemia da COVID-19;

- **Eliminazione di variabili altamente correlate:** tramite l'analisi della correlazione (Spearman[3.3]) su un campione casuale del dataset, vengono individuate ed eliminate le variabili ridondanti che mostrano un'elevata correlazione con altre feature (superiore a una soglia predefinita), in modo da prevenire fenomeni di multicollinearità e rendere più stabile e interpretabile il modello finale.

Grazie a questa sequenza di operazioni, il dataset che ne risulta è caratterizzato da una struttura compatta, priva di ridondanze e presenta le condizioni ottimali per essere sottoposto alle successive fasi.

Inoltre, prima di procedere con le ulteriori trasformazioni, il dataset-già suddiviso nei sottoinsiemi di *train*, *validation* e *test*-viene salvato, così da poter essere impiegato nella successiva fase di generazione delle regole.

3.2.4 Trasformazione dei dati

A valle delle fasi di pulizia e di suddivisione in sottoinsiemi (*train*, *validation*, *test*), il dataset viene sottoposto a una fase di trasformazione mirata ad assicurare la massima qualità informativa delle variabili e la piena compatibilità con gli algoritmi di machine learning utilizzati.

Il processo si sviluppa come di seguito descritto:

- **Allineamento delle variabili tra i sottoinsiemi:** vengono mantenute solo le colonne comuni tra gli split, ed eliminate eventuali variabili non condivise, al fine di evitare disallineamenti e problemi di consistenza nei dati in ingresso ai modelli;

[3.3]La correlazione di Spearman è una misura non parametrica di dipendenza monotona tra due variabili. È spesso preferita alla correlazione di Pearson quando i dati non seguono una distribuzione normale.

- **Rimozione delle colonne residue non informative:** ulteriori filtri permettono di eliminare colonne costanti o con valori completamente mancanti, assicurando che le feature utilizzate siano effettivamente informative e variabili;
- **Selezione delle variabili rilevanti tramite IV:** viene applicata una procedura di selezione automatica delle feature basata sull'*Information Value*[3.4], che consente di trattenere soltanto le variabili maggiormente predittive rispetto al target di rischio;
- **Trasformazione delle variabili categoriche con Weight of Evidence (WoE):** per tutte le variabili categoriche selezionate, viene applicata la trasformazione *Weight of Evidence*[3.5] [5], tecnica largamente impiegata nei contesti credit scoring per garantire la robustezza statistica e l'interpretabilità delle relazioni tra feature e rischio;
- **Costruzione del dataset finale per il training:** le variabili numeriche e categoriche trasformate vengono concatenate per formare il dataset definitivo, completo delle colonne di interesse e delle etichette di destinazione.

Il dataset finale risulta, quindi, ottimizzato per le successive fasi di addestramento e validazione dei modelli predittivi.

3.2.5 Statistiche del dataset dopo il preprocessing

Al termine delle fasi di caricamento, pulizia, trasformazione e feature engineering, il dataset risulta composto da un totale di `1232695` osservazioni, suddivise nei tre sottoinsiemi *train*, *validation* e *test* secondo la proporzione definita. La Tabella 3.1 riporta la distribuzione dei campioni e delle classi (*Buon creditore* e *Cattivo creditore*) per ciascuno split.

Tabella 3.1: Distribuzione del dataset dopo il preprocessing e la suddivisione in sottoinsiemi.

Split	Campioni totali	Buoni (0)	Cattivi (1)	% Cattivi
Train	760 416	737 412	23 004	3.03%
Val	190 105	184 280	5 825	3.06%
Test	282 174	270 898	11 276	4.00%

[3.4] L'Information Value (IV) è una metrica usata per valutare la capacità predittiva di una variabile rispetto a una variabile target binaria. Valori più alti indicano maggiore potere discriminante.

[3.5] Il Weight of Evidence (WoE) è una tecnica di trasformazione delle variabili categoriche che quantifica la forza del legame tra ciascuna modalità della variabile e il target binario.

Si osserva come, in ciascuno split, la percentuale di *cattivi creditori* rimanga compresa tra il 3% e il 4% del totale. Questo conferma la presenza di un forte sbilanciamento del dataset, come già discusso in dettaglio nella Section 2.7.

Si precisa che, al termine della fase di preprocessing, il modello è stato addestrato considerando l'intero insieme delle variabili disponibili, senza effettuare un'esplicita selezione delle feature. L'importanza relativa delle variabili è stata successivamente valutata tramite metodi di explainability con SHAP, che hanno permesso di evidenziare le feature più rilevanti nelle decisioni del sistema multiagente, come discusso nel Capitolo 4.

3.3 Addestramento del modello LightGBM e generazione dell'explainer SHAP

A partire dal dataset trasformato e ottimizzato, il modello predittivo LightGBM [17] è stato addestrato. Il processo di addestramento prevede una fase di ottimizzazione automatica degli iperparametri tramite tecniche di cross-validation[3.6] e ricerca su spazio parametrico, al fine di massimizzare le prestazioni predittive del modello.

Al termine dell'addestramento, viene inoltre costruito un explainer basato sulla libreria SHAP (*SHapley Additive exPlanations*), utilizzato per valutare l'importanza delle feature e fornire spiegazioni locali e globali sulle decisioni prese dal modello. L'explainer SHAP viene generato e salvato a partire dal modello LightGBM addestrato e dal dataset di training, rendendo così possibile una successiva fase di interpretazione, validazione e raffinamento delle regole decisionali.

3.4 Preprocessing avanzato per la generazione delle regole

Per consentire la generazione di regole interpretabili tramite l'algoritmo BRCG, è stato necessario predisporre un preprocessing avanzato e specifico, finalizzato alla creazione di dataset compatibili con tale approccio.

Il processo si articola nelle seguenti fasi:

- **Creazione delle colonne target:** a partire dal dataset salvato nella versione precedente alla trasformazione per l'addestramento, e utilizzando il modello LightGBM già addestrato, vengono calcolati per ciascun campione sia il punteggio di probabilità di default (`PD_score`) sia la relativa classe binaria (`rule_brcg`).

[3.6] La cross-validation è una tecnica statistica per stimare la performance di un modello suddividendo il dataset in sottoinsiemi, usati ciclicamente per addestramento e validazione. Aiuta a prevenire l'overfitting.

Quest'ultima viene ottenuta applicando una soglia di classificazione fissata a 0,76 (selezionata come descritto in sezione 2.6): i clienti con probabilità superiore sono considerati ad alto rischio e quindi classificati come cattivi creditori;

- **Gestione dei valori mancanti e delle categorie non viste:** viene utilizzata una strategia di gestione dei valori assenti. In particolare, le colonne che presentano una quota significativa di dati mancanti vengono rimosse dal dataset, al fine di evitare distorsioni e garantire la robustezza delle analisi successive. Per le colonne che presentano soltanto un numero limitato di valori mancanti, si procede invece con l'imputazione: i valori assenti sono sostituiti con *'MISSING'* per le variabili categoriche e con zero per quelle numeriche. Inoltre, eventuali colonne presenti nell'insieme di dati finalizzati alla fase di test ma non osservate in fase di addestramento vengono eliminate, così da assicurare coerenza tra i diversi sottoinsiemi e prevenire errori nelle fasi successive;

- **Binarizzazione delle feature:** tutte le variabili esplicative vengono trasformate in formato binario tramite il modulo *FeatureBinarizer* della libreria AIX360, sviluppata da IBM[3.7] [2]. Questa trasformazione rende il dataset nel formato adeguato per l'utilizzo dell'algoritmo BRCG;

- **Salvataggio dei dataset pronti all'uso:** i dataset così ottenuti (sia train che test), sia in versione binarizzata si come valori target associati, vengono salvati in formato serializzato[3.8], pronti per la successiva fase di addestramento e generazione delle regole.

3.5 Generazione delle regole

La generazione delle regole decisionali interpretabili è stata realizzata tramite l'algoritmo BRCG (*Boolean Rule Column Generation*) [10], applicato ai dati binarizzati ottenuti dal preprocessing avanzato. Per garantire generalizzabilità e bilanciamento delle regole apprese - dato anche il forte squilibrio tra le classi presente nel dataset (come descritto nella Sezione 2.7) - l'addestramento è stato condotto su più fold costruiti in modo bilanciato tra esempi positivi e negativi. Questa strategia si è resa necessaria sia per gestire i limiti di memoria RAM disponibili (si veda anche il Capitolo 4), sia per massimizzare la capacità di generalizzazione del sistema. Ogni fold contribuisce con

[3.7] La libreria AIX360 (AI Explainability 360) è un toolkit open-source sviluppato da IBM per supportare l'adozione di tecniche di Explainable AI. Il modulo `FeatureBinarizer` consente di trasformare le feature in rappresentazioni binarie adatte alla generazione di regole logiche.

[3.8] I file serializzati (ad esempio in formato `.pkl` o `.joblib`) permettono di salvare oggetti Python complessi in forma binaria per un caricamento veloce in sessioni successive.

una regola in formato DNF[3.9] (Disjunctive Normal Form) che, una volta combinate, consentono di ricavare informazioni rilevanti sull'intero dataset di training riguardo l'individuazione della classe target.

3.5.1 Divisione in fold bilanciati e training dei modelli BRCG

Il dataset binarizzato viene suddiviso in un elevato numero di fold, ciascuno dei quali è composto da blocchi di campioni positivi e negativi di uguale dimensione (circa 20 000 per ciascuna classe, per un totale di circa 40 000 righe a fold). La suddivisione in fold è progettata per garantire che, l'intero dataset venga coperto senza duplicazioni e con la massima rappresentatività possibile di entrambe le classi target.

Per ogni sottoinsieme:

- viene selezionato un blocco di esempi negativi (*cattivi creditori*) tramite campionamento randomizzato, ricampionando il pool a ogni iterazione per assicurare la variabilità tra i fold;
- viene affiancato un blocco contiguo di esempi positivi (*buoni creditori*), selezionati senza ricampionamento per garantire una copertura uniforme della classe positiva.

Su ciascun fold così composto viene addestrato un modello BRCG, che produce una regola interpretabile esplicitata tramite l'explainer associato al modello. Per ogni fold, oltre alle regole, vengono salvati il modello addestrato, l'explainer e un report di classificazione contenente le principali metriche di valutazione.

Questa strategia permette di:

- Ottenere regole robuste rispetto alla variabilità del campionamento e della distribuzione delle classi;
- Limitare l'overfitting su specifici sottogruppi di dati;
- Agevolare il successivo consolidamento delle regole più efficaci.

3.5.2 Aggregazione e consolidamento delle regole

Al termine della fase di generazione distribuita delle regole tramite BRCG su fold bilanciati, si procede a un'operazione di aggregazione e consolidamento volta a costruire una Knowledge Base di regole compatta, non ridondante e robusta.

Il processo è così sviluppato:

[3.9] Ogni fold è un sottoinsieme del dataset usato in una procedura di validazione incrociata. La generazione di una regola per ciascun fold consente di catturare pattern differenti e migliorare la robustezza delle regole aggregate.

- **Raccolta delle regole:** vengono estratte le regole generate provenienti da file separati per ogni fold. Ogni regola viene annotata con il fold di provenienza, così da poter tracciare la sua frequenza e la sua diffusione tra i diversi subset di training;

- **Associazione delle metriche di performance:** per ciascuna regola, viene calcolato l'F1-score medio, ricavato aggregando le metriche di valutazione ottenute nei diversi fold in cui la regola compare. Questo consente di selezionare e ordinare le regole in base alla loro efficacia predittiva;

- **Deduplicazione e ordinamento:** le regole duplicate vengono eliminate e si tiene traccia del numero di fold in cui ciascuna regola risulta attiva. Le regole così ottenute vengono poi ordinate decrescentemente in base all'F1-score medio, privilegiando quelle con migliori prestazioni e maggiore ricorrenza;

- **Esportazione e salvataggio:** l'insieme finale delle regole consolidate viene esportato in formato testuale per facilitare la lettura e il passaggio al Large Language Model.

In questo modo si ottiene un insieme di regole interpretabili, selezionate sulla base della loro robustezza e capacità discriminante, pronto per essere utilizzato dagli agenti della pipeline e che costituiscono la base conoscitiva del sistema di decisione automatica.

3.6 Architettura agent-based

La presente sezione introduce l'architettura agent-based implementata per la valutazione automatica del rischio creditizio, incentrata su una pipeline composta da due agenti: un agente predittore basato su Large Language Model e un agente validatore che integra la logica LLM con le valutazioni del modello di Machine Learning LightGBM, in linea con i più recenti sviluppi e sfide dei sistemi multi-agente discussi in letteratura [13].

Il flusso ha inizio dal profilo di un singolo cliente, che viene fornito simultaneamente all'agente predittore e al modello LightGBM. L'agente predittore produce una classificazione binaria (buon/cattivo creditore) accompagnata da una motivazione esplicita.

Parallelamente, il modello LightGBM calcola la probabilità di default del cliente, restituendo una stima probabilistica.

Le due valutazioni vengono, quindi, inoltrate all'agente validatore, che si occupa di confrontare gli esiti. Se i modelli sono concordi, la predizione viene confermata; in caso di discordanza, l'agente validatore attiva una procedura di aggiornamento delle regole, per affinare la knowledge base e migliorare le performance future.

Un aspetto centrale di questa architettura è la gestione dei dati imputati: sia l'agente predittore sia il validatore escludono sistematicamente i dati che erano stati imputati esclusivamente per consentire la generazione delle regole tramite BRCG.

In questo modo, si garantisce che il modello LLM analizzi unicamente i dati disponibili per ciascun cliente, evitando la rigidità tipica dei modelli ML e prevenendo il rischio di utilizzare informazioni fittizie durante la fase di inferenza.

Questa soluzione assicura una migliore generalizzazione e valorizza la flessibilità dell'agente LLM, che può adattarsi anche a situazioni in cui alcune informazioni non sono disponibili, superando così i limiti degli approcci basati su input/output rigidamente predefiniti[3.10].

Dal punto di vista operativo, la pipeline agent-based viene inizialmente utilizzata sui dati di validazione, permettendo l'aggiornamento dinamico delle regole e l'adattamento della knowledge base. Successivamente, la pipeline viene applicata su dati di test, sui quali vengono esclusivamente effettuate predizioni senza ulteriori aggiornamenti delle regole, così da valutare la reale capacità di generalizzazione del sistema.

Le specificità tecniche e le logiche implementative dei due agenti sono approfondite nelle Sottosezioni 3.6.1 e 3.6.2.

La logica generale di questa architettura agent-based è illustrata schematicamente in Figura 3.2.

[3.10] I modelli classici di ML richiedono che l'input contenga sempre tutte le variabili previste in fase di addestramento. I LLM, invece, possono adattarsi dinamicamente a input parziali grazie alla loro natura contestuale.

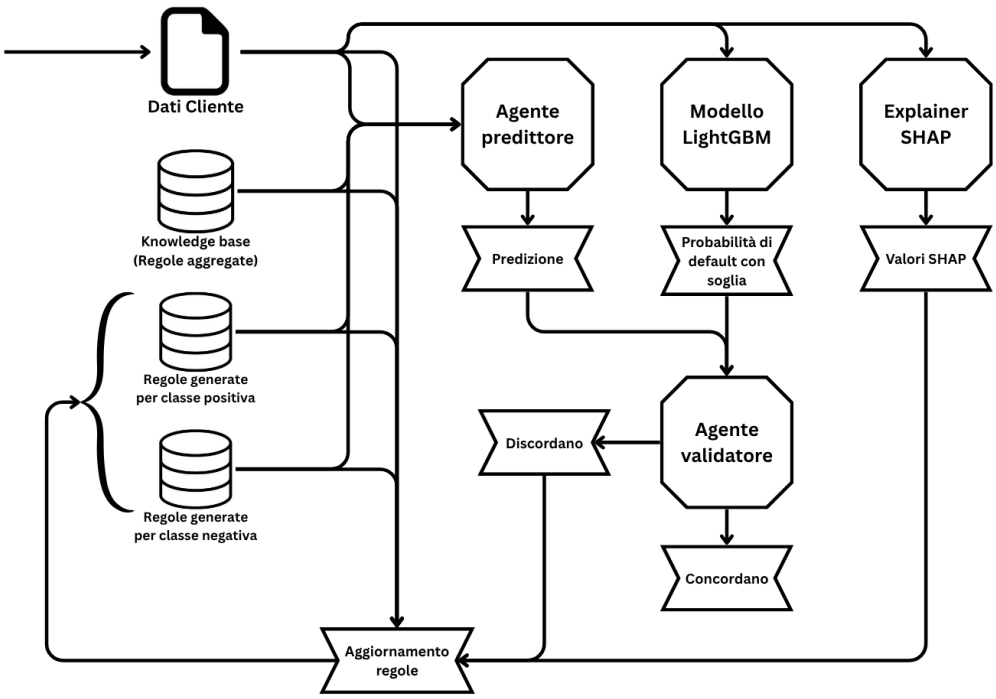

Figura 3.2: Schema della pipeline agent-based per la valutazione automatica del rischio: l'agente predittore LLM, la validazione tramite modello ML e l'aggiornamento dinamico delle regole tramite explainability SHAP.

3.6.1 Agente predittore

L'agente predittore rappresenta il primo snodo decisionale della pipeline agent-based ed è implementato tramite un Large Language Model (LLM). Questo agente riceve in input:

- **Dati del cliente**, opportunamente depurati da ogni valore imputato, così da utilizzare esclusivamente le informazioni realmente disponibili ed evitare l'utilizzo di dati fittizi;
- **Knowledge base delle regole aggregate**, ottenute tramite BRCG sull'intero training set;
- **Regole personalizzate**, sviluppate dal validatore, distinte per cattivi e buoni creditori;
- **Descrizione semantica delle feature** che caratterizzano il profilo del cliente;
- **Le dieci feature SHAP più influenti** sull'intero dataset (come dettagliato nella Sezione 2.4).

Sfruttando tutte queste fonti informative, l'agente predittore genera per ciascun cliente una predizione binaria -*Buon creditore* oppure *Cattivo creditore* - seguendo un

output strutturato e motivato. In particolare, la risposta generata dal modello LLM presenta:

- l'etichetta assegnata al cliente (classe di rischio);
- un elenco dettagliato dei **punti di forza**, cioè le feature e i motivi che hanno spinto l'agente verso la decisione presa;
- un elenco dei **punti di debolezza**, ovvero le criticità individuate che avrebbero potuto suggerire una classificazione diversa.

Per garantire la robustezza operativa anche in presenza di eventuali instabilità nella connessione ai servizi LLM, l'agente predittore implementa inoltre una gestione automatica dei tentativi di riconnessione: in caso di errore di comunicazione o timeout con il modello, il sistema effettua più tentativi progressivi di richiesta, applicando un meccanismo di *backoff*[3.11] tra un tentativo e l'altro. Solo in caso di fallimento reiterato su tutti i tentativi previsti viene generato un errore bloccante e viene interrotta l'elaborazione.

Questa modalità di output, vincolata a una struttura argomentativa esplicita (attraverso una formattazione predefinita e la distinzione tra punti di forza e di debolezza), è finalizzata a massimizzare la trasparenza, l'interpretabilità e la spiegabilità delle decisioni, evitando gli effetti "black box" tipici dei modelli ML tradizionali.

3.6.2 Agente validatore

L'agente validatore rappresenta il secondo snodo critico della pipeline agent-based e svolge il duplice compito di validazione incrociata delle predizioni e di aggiornamento dinamico della knowledge base delle regole.

Validazione delle predizioni. L'agente validatore riceve in input la predizione binaria generata dall'agente predittore e la probabilità di default restituita dal modello LightGBM per lo stesso cliente. L'output del modello ML viene elaborato tramite una soglia con valore di 0.7621: se la probabilità supera questa soglia, il cliente è classificato come "Cattivo creditore", altrimenti come "Buon creditore", come già descritto in sezione 3.4

L'agente procede quindi al confronto tra la classificazione fornita dal predittore LLM e quella prodotta da LightGBM:

- **se i risultati coincidono**, la decisione viene confermata senza ulteriori azioni.

[3.11] Il *backoff* è una strategia di attesa progressiva tra tentativi successivi di connessione o richiesta, in cui l'intervallo di tempo tra un tentativo e il successivo viene aumentato a ogni errore, tipicamente in modo esponenziale, al fine di ridurre il rischio di congestione o sovraccarico del sistema.

- **se i risultati differiscono**, il comportamento dell'agente dipende dalla fase operativa:
  - *durante la validazione*, viene avviata la procedura di aggiornamento regole;
  - *durante il test*, l'agente si limita a segnalare la discordanza, senza modificare la knowledge base.

Aggiornamento dinamico delle regole. La procedura di aggiornamento regole, attivata solo nella fase di validazione e solo in caso di discordanza, è articolata secondo una logica ben definita e utilizza una molteplicità di fonti informative:

- **Dati del cliente**, privi di valori imputati, per coerenza con quanto visto nel predittore;
- **Valori SHAP più influenti** per quella specifica predizione, calcolati in funzione della classe corretta secondo il modello ML, utilizzando l'explainer SHAP ;
- **Knowledge base delle regole aggregate**, ottenute dal processo BRCG sull'intero training set;
- **Regole personalizzate**, distinte per buoni e cattivi creditori, generate dinamicamente dal validatore;
- **Descrizione delle feature** relative al profilo cliente;
- **Probabilità di default calcolata da LightGBM**;
- **Le dieci feature SHAP più influenti a livello globale**.

Sulla base di questi elementi, l'agente validatore costruisce una richiesta strutturata all'LLM, finalizzata a generare un insieme ristretto di nuove regole (tipicamente 2 o 3[3.12]), pensate per affinare la classificazione della classe corretta. Le nuove regole proposte vengono poi sottoposte a una procedura di controllo automatizzata:

- per ciascuna nuova regola, si verifica se la stessa feature è già coperta da una regola nella knowledge base globale: in tal caso la nuova regola viene ignorata, per evitare ridondanze e conflitti;
- se la feature è già oggetto di una regola preesistente per la stessa classe, la vecchia regola viene sostituita dalla nuova versione generata;
- in tutti gli altri casi, la nuova regola viene aggiunta all'insieme delle regole validate per la classe specifica, evitando duplicati e mantenendo una struttura coerente e aggiornata della knowledge base.

[3.12] La scelta di un numero ristretto di regole è motivata dall'intenzione di mantenere la knowledge base interpretabile, evitare overfitting e garantire l'efficienza computazionale del sistema.

Questa logica consente di evolvere e rafforzare progressivamente la knowledge base del sistema, riducendo le discrepanze tra modello ML e agente LLM, migliorando l'allineamento e la coerenza delle predizioni, e favorendo la capacità del sistema di adattarsi a nuovi pattern o anomalie emergenti nei dati reali.

3.6.3 Prompt engineering e progettazione dei messaggi

La progettazione dei prompt costituisce l'elemento cardine della pipeline agent-based, influenzando in modo diretto la qualità, la struttura e la spiegabilità delle risposte generate dagli LLM sia nella fase di predizione sia nell'aggiornamento delle regole.

Scelte progettuali e motivazioni. Le strategie di prompt engineering adottate in questo lavoro si fondano su alcune best practice emerse dalla letteratura recente [11, 24, 22] e sono state ottimizzate tramite sperimentazione:

- **Vincoli formali e linguistici:** Tutti i prompt sono basati su template rigidamente strutturati, con istruzioni esplicite su formato, lingua, delimitatori e organizzazione dei blocchi, per garantire risposte uniformi e facilmente estraibili [11, 24];
- **Controllo dell'input:** Ogni messaggio verso l'LLM trasmette solo le informazioni strettamente necessarie (regole globali e personalizzate, profilo del cliente, descrizioni delle feature, valori SHAP locali e globali), serializzate con header distinti e sintassi compatta, per minimizzare il rischio di allucinazioni o ambiguità [24];
- **Separazione tra system prompt e user prompt:** Si distingue tra istruzioni di sistema e richieste utente, come raccomandato dalle migliori pratiche di prompt engineering [11, 22], per indirizzare il modello verso il comportamento desiderato;
- **Minimizzazione dell'ambiguità e del bias:** L'adozione di delimitatori standard[3.13] e istruzioni imperative ("Scrivi solo...", "Non aggiungere...", "Segui esattamente questo formato") riduce la variabilità sintattica e semantica delle risposte [11, 24];
- **Gestione della lingua nei prompt:** Pur operando in un contesto italiano, i prompt sono generalmente formulati in inglese, ma la risposta viene richiesta in italiano semplice e chiaro. Questa scelta è supportata da recenti risultati sperimentali, che dimostrano come gli LLM raggiungano la massima robustezza e coerenza quando il prompt è fornito in lingua inglese [20].

Prompt per la predizione. Il prompt costruito per il task di predizione contiene, in sequenza, tutti i blocchi informativi rilevanti per il cliente (regole BRCG, regole custom,

[3.13] L'uso di delimitatori standardizzati, come "- - -" o intestazioni fisse, è una pratica raccomandata nel prompt engineering per rendere le risposte più uniformi e facilmente interpretabili.

profilo dati depurato, descrizioni delle feature, ranking globale SHAP) e si conclude con una serie di istruzioni dettagliate sulle modalità per fornire la risposta. Si richiede espressamente che il modello risponda:

- riportando solo l'etichetta ("Buon creditore" o "Cattivo creditore") come prima riga;
- separando chiaramente "Punti di forza" e "Punti di debolezza" con i relativi motivi sintetici, usando sempre la stessa sintassi (`nome_feature: spiegazione breve`);
- utilizzando la lingua italiana in modo chiaro e semplice, senza commenti, sommari o riferimenti a regole, né variazioni nel formato.

Viene inoltre specificato di ignorare i valori imputati (valori numerici pari a 0.0), di privilegiare le feature globalmente più robuste e di evitare spiegazioni ridondanti.

Prompt per l'aggiornamento delle regole. Per la generazione di nuove regole, il prompt fornisce:

- regole base (non modificabili), regole personalizzate, profilo dati e descrizioni feature;
- SHAP locali (le feature più influenti per il caso specifico, con valore e segno), e il ranking globale delle feature;
- la classe "corretta" assegnata dal modello ML, che guida la generazione di nuove regole per la classe target.

Si istruisce l'LLM a proporre da 1 a 3 nuove regole booleane (ognuna con almeno due condizioni AND), privilegiando feature globalmente robuste e formulazioni generalizzabili Si impone di rispondere solo con la lista di regole, senza spiegazioni, commenti o variazioni di formato, per facilitare la verifica automatica e l'unione delle regole nella knowledge base.

3.7 Integrazione delle tecniche di explainability nella pipeline

L'integrazione di tecniche di explainability all'interno della pipeline progettata pone la spiegabilità come parte integrante e attiva del ciclo decisionale.

In particolare, la pipeline sfrutta la libreria SHAP per produrre spiegazioni sia globali sia locali sulle predizioni del modello LightGBM. I valori SHAP globali vengono

utilizzati per identificare le feature mediamente più influenti sull'intero dataset, mentre i valori SHAP locali consentono di individuare, per ciascun cliente, quali caratteristiche abbiano maggiormente inciso sulla valutazione di rischio.

Questa doppia prospettiva è essenziale nella fase di validazione e nella logica di aggiornamento dinamico della knowledge base delle regole. Infatti, in presenza di discordanza tra la predizione dell'agente LLM e quella del modello ML, il validatore utilizza i valori SHAP locali e globali per individuare e suggerire quali siano le feature effettivamente più rilevanti nella predizione; l'aggiornamento delle regole avviene però sempre sulla base dei dati reali e osservati del cliente, così da garantire che ogni nuova regola rifletta fedelmente il caso concreto.

In tal modo, l'explainability fornisce una spiegazione delle decisioni già prese e contribuisce in modo diretto a rendere il sistema adattivo e trasparente, permettendo l'evoluzione dinamica delle regole sulla base dei pattern effettivamente rilevati nei dati. Questa integrazione della spiegabilità nel ciclo decisionale è un fattore abilitante per il miglioramento continuo delle prestazioni predittive e per il rispetto dei requisiti di tracciabilità e verificabilità che caratterizzano le applicazioni di intelligenza artificiale nel settore finanziario.

3.8 Flusso dati e automazione della pipeline

L'automazione e la strutturazione del flusso dati rappresentano un aspetto fondamentale per garantire la robustezza e la riproducibilità del sistema, elementi particolarmente rilevanti nell'ottica di una futura integrazione in ambienti di produzione aziendale.

La pipeline è organizzata secondo una sequenza di step modulari e automatizzati:

1. **Caricamento e preparazione dei dati:** i dati raw vengono letti da file *parquet*, integrati e tipizzati automaticamente;

2. **Feature engineering e pulizia:** la creazione di nuove variabili e la pulizia dei dati avvengono tramite script dedicati, con parametri configurabili;

3. **Addestramento e explainability:** l'addestramento del modello LightGBM e la generazione degli explainer SHAP sono eseguiti in modalità batch, con salvataggio automatico dei modelli e delle metriche;

4. **Preprocessing per regole interpretabili:** la generazione dei dataset binarizzati, necessari per l'estrazione di regole BRCG, è completamente automatizzata, con gestione sistematica dei dati mancanti e dei valori out-of-sample;

5. **Estrazione e aggregazione delle regole:** la generazione dei fold bilanciati, l'addestramento dei modelli BRCG e l'aggregazione delle regole vengono orchestrate tramite script sequenziali e ripetibili;

6. **Pipeline agent-based:** l'intera logica di interazione tra agente predittore, validatore e sistema di aggiornamento regole è gestita tramite script modulari, che permettono validazione incrementale e valutazione finale su dati di test mai visti dal sistema precedentemente.

Il flusso dati è stato progettato per essere totalmente automatizzato, minimizzando l'intervento manuale e garantendo la ripetibilità degli esperimenti, la tracciabilità di ogni passaggio (tramite salvataggi intermedi) e la semplicità di estensione o adattamento a nuovi dataset o nuovi moduli.

Capitolo 4

Dettagli implementativi e sviluppo

4.1 Tecnologie adottate e ambiente di sviluppo

Lo sviluppo del progetto si è svolto in un contesto ibrido, alternando risorse locali e infrastrutture cloud. La scelta di tale architettura si è rivelata fondamentale a causa dell'ampiezza del dataset utilizzato e della complessità delle procedure di addestramento e validazione dei modelli di intelligenza artificiale impiegati.

4.1.1 Hardware locale

Le prime fasi di sviluppo sono state condotte su un notebook Dell con processore Intel® Core™ i7-1185G7 (11^{th} Gen) e 16 GB di RAM. La scheda grafica integrata (Intel Iris Xe Graphics) non è stata sfruttata per l'addestramento o l'esecuzione di modelli di machine learning. Il sistema operativo utilizzato è Windows 11 Pro a 64 bit.

4.1.2 Cloud Computing e AWS SageMaker

Le attività che richiedevano un'elevata potenza di calcolo, in particolare il caricamento del dataset e la rispettiva procedura di pulizia e trasformazione dati e, ancor di più, la generazione delle regole con l'algoritmo BRCG che include la risoluzione di problemi di ottimizzazione - per cui abbiamo utilizzato il solver ECOS[4.1] - sono state gestite attraverso Amazon SageMaker, in ambiente JupyterLab. È stata utilizzata un'istanza di tipo `ml.m5d.16xlarge`, dotata di 64 vCPU, 256 GB di RAM e storage NVMe ad alte prestazioni. L'utilizzo del cloud si è rivelato imprescindibile per la gestione e in particolare l'elaborazione del dataset e per l'esecuzione di task paralleli, ad esempio, la generazione dei report di classificazione per ciascun fold, eseguita contemporaneamente su tutte le partizioni. Questa soluzione ha permesso di superare i limiti fisici della

[4.1] ECOS (Embedded Conic Solver) è un risolutore per problemi di programmazione conica, in particolare SOCP (Second-Order Cone Programming). Documentazione disponibile su: `https://github.com/embotech/ecos/wiki/Usage-from-Python`.

macchina locale e di ridurre sensibilmente i tempi di calcolo, ottimizzando al tempo stesso i costi computazionali attraverso un uso mirato delle risorse cloud.

4.1.3 Ambiente di sviluppo software

Lo sviluppo del codice è avvenuto principalmente tramite Visual Studio Code. L'intero progetto è stato versionato tramite il repository GitLab aziendale e, per garantire la massima riproducibilità degli esperimenti e l'isolamento dell'ambiente, è stato predisposto un ambiente virtuale `conda` (`tesi_env`), con tutte le dipendenze gestite tramite il file `requirements.txt`. Il linguaggio utilizzato per tutti gli script è Python, in versione 3.10.18.

4.1.4 Librerie e framework utilizzati

La soluzione progettata si è avvalsa di varie librerie, selezionate in funzione delle specifiche esigenze. Le principali dipendenze includono:

- **Data Science e Machine Learning:** `numpy`, `scipy`, `pandas`, `polars`, `scikit-learn`, `joblib`;
- **Modelli ML e Explainability:** `lightgbm`, `shap`, `AIX360` (IBM Explainability 360), `optuna`;
- **Ottimizzazione:** `cvxpy`, `ecos`;
- **AI generativa:** `openai`, `tiktoken`;
- **Visualizzazione e analisi:** `matplotlib`, `seaborn`;
- **Web e backend:** `streamlit`, utilizzato per la demo interattiva.

4.1.5 Controllo delle versioni

Tutte le attività di sviluppo, monitoraggio e collaborazione sono state facilitate dal repository GitLab aziendale. Il branch principale, `f_diliberti`, ha raccolto progressivamente i contributi sviluppati in branch secondari e successivamente integrati tramite merge, seguendo una metodologia simile al *feature branching*[4.2].

Nelle fasi cloud, i dati sono stati temporaneamente archiviati su storage JupyterLab (integrato in SageMaker) e successivamente trasferiti in locale per le ulteriori elaborazioni e validazioni, una volta conclusi i task computazionalmente più onerosi. La

[4.2] Il *feature branching* è una strategia di versionamento Git in cui ogni nuova funzionalità viene sviluppata in un branch separato, per poi essere integrata nel branch principale tramite *merge*. Descrizione disponibile su: `https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow`.

sincronizzazione e il trasferimento dei dati e degli script tra ambiente cloud e locale sono stati resi agevoli grazie all'utilizzo di Git, che ha permesso di mantenere allineate le versioni del codice.

4.1.6 Scelta e configurazione del modello LLM

Per tutte le interazioni con l'API OpenAI, il sistema è stato configurato per utilizzare il modello `gpt-4.1-mini-2025-04-14`. Questa versione rappresenta la soluzione più bilanciata tra qualità delle risposte, velocità di esecuzione e sostenibilità economica, tra i modelli di OpenAI al momento disponibili.

La scelta è stata adeguatamente ponderata a seguito di un confronto con altri modelli disponibili: il modello completo `gpt-4.1`, sebbene più potente, presentava costi significativamente superiori; la versione `gpt-4.1-nano`, invece, pur essendo più economica, non garantiva la stessa affidabilità e capacità di ragionamento nella valutazione creditizia.

Per questo motivo, `gpt-4.1-mini` è stato adottato come modello standard per tutti gli agenti LLM della pipeline, assicurando un comportamento stabile e riproducibile durante tutte le fasi di addestramento, validazione e test.

4.2 Organizzazione del progetto e struttura dei file

L'organizzazione del progetto è stata concepita per garantire la modularità e la facilità di manutenzione del codice, sia durante lo sviluppo che nelle successive fasi di validazione e test. La struttura segue le buone prassi già consolidate nello sviluppo di soluzioni di machine learning e intelligenza artificiale, separando in modo chiaro la logica applicativa dai dati, dagli script di addestramento e dai prodotti di output.

Il progetto principale, `loan-and-mortgage-evaluation`, si articola in diverse macro-aree:

- **Directory dei dati** (`data/`): ospita il dataset utilizzato e scomposto nei diversi file che lo compongono e i dataset generati nelle varie fasi del progetto, sia in formato grezzo sia preprocessato. All'interno sono presenti sottocartelle dedicate ai risultati dell'analisi del dataset, ai dati di addestramento, validazione e test, ai risultati degli addestramenti di BRCG (`brcg_output/`), ai modelli addestrati (ad esempio, i file `.pkl` di LightGBM), ai report di classificazione, alle definizioni delle feature e agli output delle metriche di valutazione. Questa suddivisione consente di tenere traccia delle diverse versioni dei dati e degli artefatti generati nei vari step della pipeline.

- **Codice sorgente** (`src/`): contiene tutti gli script Python relativi all'implementazione della pipeline, suddivisi in moduli logici e sottopacchetti. In particolare:
  - `src/agents/`: raccoglie gli agenti principali (predictor e validator), responsabili della predizione e validazione basata sulle regole, tramite LLM e modelli ML;
  - `src/BRCG/`: include gli script per la preparazione dei dati, la generazione in fold e la fusione delle regole interpretabili con l'algoritmo BRCG;
  - `src/scorecard_modified/`: contiene i moduli di caricamento dataset, feature engineering, data cleaning e addestramento del modello LightGBM; si tratta di codice proveniente da un altro progetto di tesi, modificato e adattato al presente lavoro[1];
  - `src/utils/`: ospita le utility trasversali (gestione I/O, prompt per LLM, caricamento delle regole, ecc.);
  - altri script di alto livello, quali `prepare_dataset_and_model.py` e `run_agents_pipeline.py`, che orchestrano le principali pipeline di training e validazione, oltre allo script di analisi esplorativa.
- **Documentazione e metadati**: la cartella base del progetto include file come `README.md`, indispensabile per la presentazione e la prima configurazione, e il file `requirements.txt` per la gestione delle dipendenze.
- **Gestione delle credenziali**: una directory apposita (`credential/`) contiene file non tracciati da Git, caricati all'occorrenza come variabili d'ambiente tramite Python[4.3], come le chiavi di accesso fornite dall'azienda per le richieste ai modelli di OpenAI (`openai_key.env`), a garanzia della sicurezza delle informazioni sensibili.
- **Repository esterni**: la presenza della sottocartella `AIX360/` indica l'integrazione di componenti esterni (in questo caso, la libreria IBM Explainability 360 per la generazione di regole interpretabili), che è stata in minima parte modificata per sopperire ad alcune differenze di versione tra le librerie dovute alla compresenza di più progetti.

La struttura adottata riflette una chiara separazione delle responsabilità, facilitando l'isolamento dei moduli, la riproducibilità degli esperimenti e l'estensione futura del progetto. Ogni componente risulta così facilmente localizzabile e manutenibile.

[4.3] L'uso delle variabili d'ambiente consente di separare le informazioni sensibili dal codice sorgente, evitando che chiavi API o credenziali vengano accidentalmente caricate su git o esposte.

4.3 Descrizione degli script principali

Il progetto si basa su una pipeline articolata, di seguito si presenta una panoramica degli script principali, esplicitando per ciascuno il ruolo, la logica e le modalità di interazione con gli altri componenti del sistema.

prepare_dataset_and_model.py

Questo script rappresenta il punto di partenza dell'intera pipeline. Il suo scopo è preparare i dati per la fase di modellazione creditizia secondo un approccio scorecard. In particolare, gestisce:

- la lettura e aggregazione dei dati grezzi da file `.parquet`;
- le operazioni di feature engineering, pulizia, trasformazione e selezione delle variabili rilevanti;
- la suddivisione dei dati in training, validation e test set, con salvataggio dei dataset in formato CSV;
- il training, tuning e salvataggio del modello di classificazione finale (LightGBM), utilizzato come base per il calcolo della probabilità di rischio di default (PD_score);
- la generazione e salvataggio dei valori di spiegabilità tramite SHAP.

prepare_brcg_dataset.py

Questo modulo è responsabile della costruzione dei dataset binarizzati necessari all'algoritmo BRCG (Boolean Rule Column Generation). Le sue principali funzioni sono:

- arricchimento dei dati con i valori di probabilità di default (PD_score) e la label binaria `rule_brcg`, derivata applicando la soglia di 0,76, scelta seguendo le modalità descritte in 2.6, sulla probabilità di default;
- gestione dei valori mancanti e delle categorie non viste nel test set;
- binarizzazione delle feature tramite la libreria AIX360 di IBM;
- salvataggio dei dataset binarizzati e delle rispettive etichette in formato `.pkl`.

generate_rules_balanced_folds.py

Questo script rappresenta il cuore della generazione di regole interpretabili secondo l'approccio BRCG. Opera secondo la logica dei fold bilanciati, in cui ciascun fold contiene pari numerosità di esempi positivi e negativi. Le sue funzionalità includono:

- training parallelo su più fold dei modelli di regole booleani (tramite la classe `BooleanRuleCG` di AIX360 e `BRCGExplainer`);

- salvataggio degli explainers, dei checkpoint e delle regole estratte per ciascun fold;
- valutazione delle regole tramite classificatori e generazione dei report di validazione;
- gestione efficiente delle risorse tramite multiprocessing e ottimizzazione del consumo di memoria.

merge_rules.py

Questo script ha lo scopo di aggregare e selezionare le regole generate sui diversi fold. In particolare:

- legge tutte le regole prodotte dai singoli fold e ne valuta la robustezza e le metriche di performance (in particolare, F1-score);
- costruisce un ranking delle regole più affidabili, aggregando le statistiche di validazione;
- salva l'elenco finale delle regole selezionate sia in formato testuale sia in CSV, rendendole disponibili per l'utilizzo da parte degli agenti LLM.

predictor_agent.py

Lo script implementa la logica dell'agente predittore basato su LLM, incaricato di esprimere una prima valutazione del profilo creditizio del cliente. La sua architettura prevede:

- il caricamento e l'aggregazione delle regole di base e delle regole personalizzate validate dal secondo agente;
- la costruzione del prompt da inviare al modello generativo, integrando descrizioni delle feature e regole specifiche per "buon creditore" o "cattivo creditore";
- l'invio della richiesta di predizione al modello LLM tramite API OpenAI, con una gestione automatica dei tentativi di riconnessione: in caso di errore di comunicazione o timeout, il sistema effettua fino a cinque tentativi progressivi (*backoff*), incrementando l'intervallo di attesa tra un tentativo e il successivo, così da garantire maggiore robustezza operativa anche in presenza di instabilità di rete; in output, vengono restituiti l'etichetta assegnata al cliente e una motivazione strutturata, articolata in punti di forza e punti di debolezza in funzione delle feature rilevanti;
- l'estrazione strutturata della risposta, restituendo la label e i punti chiave per la validazione successiva.

validator_agent.py

Il validatore costituisce l'agente di controllo e spiegabilità della pipeline. Per ogni predizione effettuata dall'agente LLM, il validator:

- confronta la predizione LLM con l'output del modello ML (LightGBM), determinando il grado di concordanza e il relativo PD_score;
- in caso di discordanza:
  - seleziona automaticamente le feature più influenti tramite i valori SHAP;
  - costruisce un prompt di validazione che sfrutta le feature SHAP e le regole pre-esistenti, chiedendo al LLM di suggerire eventuali nuove regole o modifiche a quelle esistenti;
  - aggiorna dinamicamente la knowledge base delle regole customizzate, differenziando tra regole per "buoni" e "cattivi" creditori.

run_agents_pipeline.py

Questo script ha la funzione di orchestratore della pipeline di validazione automatica. Consente di eseguire sull'intero dataset di validazione e di test:

- la validazione automatica: per ogni riga di input nel set di validazione, il sistema genera la predizione dell'agente LLM e la validazione del modello ML; è importante specificare che l'aggiornamento delle regole viene effettuato esclusivamente sulle righe in cui la predizione del modello ML coincide con il target di ground truth, così da ridurre il rumore dovuto a casi ambigui e garantire maggiore affidabilità al processo di raffinamento della knowledge base;
- la valutazione quantitativa delle prestazioni dell'intero sistema ibrido, condotta su un campione completo del set di test, tramite metriche di accuratezza, F1-score, report di classificazione e analisi dettagliata degli errori (falsi negativi/positivi).

demo.py

Infine, lo script `demo.py` realizza una demo interattiva tramite Streamlit, permettendo di esplorare e visualizzare il funzionamento del sistema su dati reali di test:

- caricamento e selezione dinamica del cliente da valutare;
- presentazione affiancata delle predizioni dell'agente LLM, del modello ML e dell'etichetta di ground truth effettiva, con evidenziazione immediata di eventuali concordanze o discordanze tra i tre risultati;
- visualizzazione chiara dei punti di forza e di debolezza individuati dall'agente LLM, confrontati con le feature a maggiore impatto secondo lo SHAP explainer;

- generazione automatica di una tabella che sintetizza la coerenza tra le motivazioni addotte dall'LLM (punti di forza/debolezza) e il segno e valore delle corrispondenti feature SHAP, segnalando esplicitamente eventuali incoerenze tra i due approcci esplicativi;
- confronto visuale tramite grafico delle top-10 feature SHAP per ciascun caso analizzato, per valutare l'importanza relativa delle variabili chiave;
- possibilità di abilitare o disabilitare l'uso delle regole validate manualmente e analizzare l'impatto di queste ultime sui risultati.

Questo strumento rappresenta una piattaforma di test avanzata e una dimostrazione interattiva delle potenzialità della soluzione, rendendo accessibili sia l'analisi delle predizioni sia la verifica puntuale della trasparenza e coerenza tra i diversi livelli di spiegazione del sistema.

4.4 Gestione degli input/output

La pipeline implementata gestisce i flussi di dati tramite una struttura di file intermedi e output organizzati all'interno delle sottocartelle della directory `data/`. I dati di partenza, in formato `.parquet` e organizzati su tre livelli di profondità, sono raccolti in `data/data_for_train/` e già dettagliati nella Sezione 2.3.

Nei principali step della pipeline vengono generati molteplici output intermedi e finali:

- **Dati preprocessati:** vengono prodotti file `.csv` relativi agli split di training, validation e test, sia nelle versioni non elaborate sia dopo pulizia e trasformazione. Questi sono collocati in `data/dataset/` con il modello LightGBM addestrato e l'explainer SHAP , mentre i dati arricchiti con PD_score e label binaria per BRCG sono salvati in `data/dataset/data_w_pd/`.
- **Dataset binarizzati e modelli:** i dati pronti per la generazione delle regole, in formato `.pkl`, sono organizzati in `data/brcg_output/`, insieme agli explainer addestrati, ai checkpoint, ai report di validazione e alle regole, sia per singolo fold che aggregate. Inoltre sono presenti i file di testo separati per le regole validate specifiche per "buoni" e "cattivi" creditori.
- **Feature engineering e analisi**: i file `.csv` con la descrizione delle feature originali e ingegnerizzate sono archiviati in `data/`, mentre i risultati delle analisi esplorative (statistiche descrittive, top SHAP, grafici) si trovano in `data/analysis/`.

- **Output di valutazione:** i report finali di classificazione, le metriche aggregate delle performance (accuracy, F1-score, analisi degli errori) e la soglia ottimale di classificazione individuata sono raccolti nella directory `data/output_metrics/`, unitamente ai report relativi alle performance del modello ML, sviluppato nell'ambito di un altro progetto di tesi [1].

Lo scambio di dati tra i diversi script avviene generalmente attraverso la lettura e scrittura di file intermedi; ciò include i dataset preprocessati, le regole, i modelli, i report e le metriche. L'accesso a dati sensibili, come la API key di OpenAI, è gestito tramite file `.env` collocati nella cartella `credentials/` e caricati all'occorrenza dagli script di utilità.

Gli output finali della pipeline vengono resi disponibili sia in forma di report strutturati (ad esempio, i file di output metrics contenenti i risultati delle valutazioni), sia tramite l'interfaccia interattiva realizzata con Streamlit nello script `demo.py`, che permette di esplorare visivamente le predizioni, le regole applicate e le spiegazioni SHAP per ogni cliente.

La gestione efficiente del caricamento e salvataggio dei file è affidata a funzioni di utilità centralizzate nello script `utils/io.py`, che assicura consistenza nei flussi di input/output tra i diversi moduli della pipeline.

4.5 Ottimizzazione, automazione e gestione risorse

L'intero sistema è stato progettato adottando principi di efficienza computazionale e gestione attenta delle risorse, al fine di massimizzare le prestazioni e contenere i costi operativi, soprattutto nelle fasi ad alto impatto computazionale.

Per le operazioni particolarmente onerose, come la generazione delle regole tramite l'algoritmo BRCG e la risoluzione di problemi di ottimizzazione con il solver ECOS (come già anticipato), è stata implementata una strategia ibrida locale/cloud. Il training parallelo dei modelli su più fold è stato affidato a un'istanza AWS SageMaker — avente le caratteristiche descritte nella Sezione 4.1.2 — riservando invece le attività meno complesse alla macchina locale. Questo approccio ha consentito di ridurre sensibilmente i costi del cloud, sfruttando le risorse on-demand solo per task realmente critiche dal punto di vista computazionale.

Un elemento chiave dell'ottimizzazione è stata la parallelizzazione delle fasi più dispendiose, in particolare nella generazione e valutazione delle regole. Grazie all'uso delle librerie `joblib` e `multiprocessing`, la produzione dei `classification_report` per ciascun fold è stata eseguita in parallelo su tutte le CPU disponibili: ciò ha permesso di completare i report per ogni fold in circa 40 minuti, mentre l'esecuzione sequenziale, secondo una stima fatta preventivamente, avrebbe richiesto oltre 21 ore complessive,

comportando un impatto economico notevole sulle risorse cloud. Analogamente, anche la generazione delle regole su fold bilanciati - pur richiedendo un tempo significativo a causa della dimensione dei dati e della complessità del solver - è stata ottimizzata tramite la suddivisione dei dati in blocchi.

La gestione della memoria e dello storage è stata anch'essa ottimizzata: la suddivisione in fold bilanciati, il caricamento e la lavorazione incrementale di porzioni di dati e il cleaning esplicito tramite `gc.collect()` hanno consentito di evitare saturazioni della RAM anche durante le fasi più intensive. Gli output intermedi (modelli, explainers, report, regole) sono stati salvati in modo incrementale, una scelta che da un lato ha inevitabilmente appesantito il repository di progetto con numerosi dati temporanei, ma dall'altro ha garantito la possibilità di riprendere rapidamente l'elaborazione in caso di interruzioni, riducendo il rischio di perdita dati e soprattutto evitando la necessità di rieseguire step computazionalmente costosi, con conseguente risparmio di tempo e risorse economiche.

Dal punto di vista dell'automazione, tutti i processi sono stati incapsulati in script modulari ed eseguibili end-to-end, riducendo la necessità di interventi manuali e garantendo la ripetibilità degli esperimenti. L'orchestrazione della pipeline, affidata a script come `run_agents_pipeline.py`, consente di processare interi dataset di validazione o test, con salvataggio automatico dei risultati e delle metriche aggregate. Nella pratica, tuttavia, `run_agents_pipeline.py` è stato eseguito principalmente su campioni rappresentativi dei dati, piuttosto che sull'intero split, per ragioni di sostenibilità economica: i dataset di validazione e test erano infatti costituiti da centinaia di migliaia di righe, e l'elaborazione completa avrebbe comportato costi di richieste OpenAI molto elevati, difficilmente giustificabili in fase sperimentale.

Le strategie adottate hanno consentito di realizzare un sistema efficiente, scalabile e adattabile a dataset di dimensioni e complessità variabili. L'ottimizzazione mirata delle risorse e l'automazione dei processi hanno garantito la possibilità di affrontare scenari sperimentali impegnativi mantenendo sotto controllo i tempi di elaborazione e i costi, grazie a un bilanciamento attento tra elaborazione locale e utilizzo del cloud.

4.6 Test e validazione del sistema

La validazione della soluzione è stata condotta seguendo una metodologia rigorosa, volta a garantire sia la correttezza formale della pipeline sia la robustezza delle procedure di predizione e spiegazione adottate. Tutte le prove sono state eseguite su campioni rappresentativi dei dati di validazione e test, con particolare attenzione all'automatizzazione e alla ripetibilità delle procedure.

La pipeline di validazione automatica, realizzata tramite lo script `run_agents_pipeline.py`, prevede per ogni osservazione l'esecuzione della predizione tramite agente predittore e

il successivo confronto con l'output del modello ML, eseguito dall'agente validatore. In caso di discordanza tra le due valutazioni, viene attivata la procedura di aggiornamento delle regole basata sulle feature SHAP più influenti, a rafforzare la trasparenza e la coerenza delle decisioni. L'intero processo è orchestrato tramite script organizzati in moduli, con salvataggio automatico dei report e delle metriche di valutazione nelle rispettive directory di output.

Per garantire la riproducibilità degli esperimenti, la pipeline utilizza un valore fisso per le operazioni randomizzate e permette il rilancio delle validazioni a partire da artefatti intermedi già generati. Oltre ai test automatici, la soluzione può essere testata manualmente tramite l'interfaccia Streamlit (`demo.py`), che consente di esaminare in modo interattivo le predizioni e le spiegazioni dei singoli casi, facilitando la verifica qualitativa dei risultati e il confronto diretto tra le motivazioni proposte dall'agente LLM e le spiegazioni SHAP.

I dettagli relativi alle metriche ottenute e alle prestazioni del sistema saranno discussi nel Capitolo 5, dedicato all'analisi dei risultati sperimentali.

4.7 Difficoltà incontrate e soluzioni adottate

La fase di sviluppo ha portato ad affrontare diverse sfide che hanno richiesto la sperimentazione di diverse strategie. Di seguito sono elencate le principali criticità riscontrate e le relative soluzioni implementate:

1. **Limiti hardware locali nella generazione delle regole BRCG:**
   La memoria RAM del computer locale risultava insufficiente per eseguire il training dei modelli sul dataset, in quanto di grandi dimensioni.
   Soluzione: Utilizzo di istanze cloud AWS SageMaker ad alta capacità (64 vCPU, 256 GB RAM) per la generazione dei modelli.

2. **Ottimizzazione dei tempi di calcolo per la valutazione su molti fold:**
   L'esecuzione sequenziale dei report di classificazione per 32 fold avrebbe richiesto oltre 21 ore.
   Soluzione: Parallelizzazione della generazione dei `classification_report` tramite `joblib` e `multiprocessing`, riducendo il tempo totale a circa 40 minuti.

3. **Gestione del rischio di saturazione della RAM:**
   La lettura simultanea di grandi porzioni di dati causava rischi di overflow.
   Soluzione: Suddivisione dei dati in blocchi/fold gestibili, caricamento selettivo e cleaning della memoria con `gc.collect()`.

4. **Persistenza e salvataggio degli output intermedi:**
   In caso di interruzione o crash, il rischio era la perdita di ore di computazione.

Soluzione: Salvataggio incrementale di modelli, explainers e checkpoint ad ogni fold/processo; gestione di alcuni dati temporanei fuori dal controllo versione per evitare appesantimento del repository.

5. **Contenimento dei costi su OpenAI:**
   L'esecuzione degli script su tutto il dataset avrebbe generato costi proibitivi, soprattutto nella fase di validazione delle regole e successiva valutazione.
   Soluzione: Esecuzione di `run_agents_pipeline.py` solo su campioni rappresentativi.

6. **Integrazione tra moduli di progetti diversi:**
   Alcuni script (scorecard, BRCG) derivavano da progetti differenti, con versioni di librerie non sempre compatibili.
   Soluzione: Refactoring e adattamento dei moduli, aggiornamento selettivo delle librerie e isolamento dell'ambiente tramite `conda`.

7. **Gestione delle regole custom buoni/cattivi:**
   La separazione tra regole per "buoni" e "cattivi" creditori non era prevista originariamente: si prevedeva un solo insieme di regole, oltre a quelle generate da BRCG, per i "cattivi" creditori. Ciò portava a risultati insoddisfacenti e numerosi errori.
   Soluzione: Ridefinizione della struttura delle regole, separazione fisica dei file e aggiornamento delle funzioni di caricamento e merge.

8. **Complessità nella binarizzazione delle feature per BRCG:**
   La presenza di feature non viste o valori anomali nel test impediva una binarizzazione consistente.
   Soluzione: Implementazione di routine di pulizia (imputazione, gestione delle unseen categories) e validazione tramite funzioni dedicate in `utils/data_utils.py`.

9. **Costruzione di fold bilanciati:**
   Sbilanciamento delle classi nella popolazione originale.
   Soluzione: Campionamento stratificato e costruzione manuale di fold con pari numero di esempi positivi e negativi, per garantire generalizzazione e ottenere regole discriminanti sull'intero dataset.

10. **Parsing delle risposte strutturate dell'LLM:**
    Formati variabili delle risposte dei modelli OpenAI rischiavano di generare errori nel parsing automatico.
    Soluzione: Regole di parsing robuste e verifica della presenza di separatori/testi chiave, secondo le tecniche di prompt engineering descritte nella Sezione 3.6.3.

11. **Sincronizzazione e trasferimento dati tra cloud e locale:**
    Necessità di mantenere allineati i file tra ambienti diversi senza perdita di informazioni.
    Soluzione: Utilizzo sistematico di GitLab per il versionamento del codice e sincronizzazione dati.

12. **Gestione delle credenziali e sicurezza:**
    Rischio di esposizione accidentale delle chiavi di accesso alle API.
    Soluzione: Gestione delle credenziali tramite file `.env` non tracciati su Git e caricati solo localmente negli script.

13. **Individuazione della soglia ottimale per la classificazione:**
    L'impostazione iniziale della soglia a 0,65 per la binarizzazione del punteggio `PD_score` ha prodotto risultati insoddisfacenti, con un elevato numero di errori e una classificazione poco discriminante tra buoni e cattivi creditori.
    Soluzione: Sviluppo di uno script dedicato per la ricerca sistematica della soglia ottimale, che ha portato a identificare il valore 0,76 come punto di equilibrio più efficace tra precision, recall e F1-score, migliorando significativamente le prestazioni del sistema.

14. **Limiti di accuratezza del modello ML rispetto al ground truth:**
    Le metriche di performance ottenute dal modello ML rispetto al ground truth non risultano pienamente soddisfacenti, comportando il rischio che le regole apprese e validate dall'agente LLM riflettano eventuali bias o limiti del modello supervisionato.
    Soluzione: Rafforzamento delle tecniche di prompt engineering e progettazione delle fasi di validazione, con l'obiettivo di favorire la generalizzazione e l'autonomia dell'agente LLM nel riconoscere pattern distintivi, potenzialmente superiori rispetto a quelli appresi dal solo modello ML.

Il superamento di tali criticità ha contribuito a consolidare un approccio iterativo e pragmatico allo sviluppo, in cui ogni ostacolo è stato trasformato in occasione di miglioramento dell'architettura e della trasparenza dell'intero sistema.

Capitolo 5

Analisi dei Risultati Sperimentali

5.1 Setup sperimentale

Gli esperimenti condotti per la valutazione della pipeline proposta si basano sui dati, sulle tecniche di pre-processing e sulle strategie di feature engineering descritte nei capitoli precedenti. In questa sezione si fornisce una sintesi del protocollo sperimentale adottato, rimandando alle sezioni specifiche per i dettagli implementativi.

5.1.1 Dataset e suddivisione dei dati

Il dataset utilizzato, descritto in dettaglio nella Sezione 2.3, è costituito da osservazioni relative a richieste di prestito e mutuo, caratterizzate da un insieme eterogeneo di variabili numeriche e categoriche. I dati sono stati sottoposti a un processo di pulizia, gestione dei valori mancanti, binning e encoding delle variabili categoriche secondo la procedura esposta nei capitoli precedenti. La suddivisione in sottoinsiemi di *training*, *validation* e *test* è stata effettuata seguendo uno schema temporale, in modo da simulare uno scenario realistico di predizione su nuove richieste.

5.1.2 Tecniche di validazione e ambiente computazionale

Le attività di generazione e validazione delle regole sono state condotte mediante addestramento su training set e validazione su validation set, senza mai utilizzare dati di test durante la fase di addestramento. Tutte le fasi computazionali sono state eseguite in ambiente Python 3.10, con il supporto delle librerie `scikit-learn`, `LightGBM`, `Polars`, `OpenAI` e altre, come specificato nella Sezione 4.1.4. Le attività computazionalmente intensive sono state eseguite su istanze cloud AWS SageMaker, secondo quanto descritto nella Sezione 4.1.2.

Si precisa che, per motivi legati ai costi computazionali e all'utilizzo delle API, le procedure di validazione e aggiornamento delle regole sono state condotte su un

campione di 500 osservazioni, estratte dallo split di validazione, facendo in modo che le classi fossero equamente rappresentate, tramite una selezione bilanciata dei campioni. Questo approccio, sebbene consenta un'analisi rappresentativa e un contenimento dei costi, potrebbe influenzare la robustezza e la generalizzabilità delle regole apprese rispetto a una validazione condotta sull'intero dataset, potenzialmente limitando la solidità delle conclusioni sperimentali.

Per ulteriori dettagli relativi alla struttura dei dati, alle tecniche di feature engineering e alle implementazioni algoritmiche, si rimanda ai capitoli precedenti. Di seguito vengono presentati i risultati sperimentali e il confronto tra le diverse architetture sperimentate.

5.2 Metriche di valutazione

La valutazione della pipeline proposta si articola su diversi livelli, al fine di fornire una misurazione completa sia delle prestazioni predittive sia dell'efficacia esplicativa del sistema. Di seguito si illustrano le metriche adottate e il loro significato nel contesto sperimentale.

5.2.1 Metriche di accuratezza e robustezza

Per la valutazione quantitativa delle prestazioni predittive, sono state adottate le metriche standard di classificazione prodotte dal `classification_report` di `scikit-learn`[5.1]. Le principali metriche considerate sono:

- **Accuracy**: la percentuale di predizioni corrette rispetto al totale dei campioni valutati;
- **Precision**: la frazione di soggetti classificati come positivi che risultano effettivamente tali;
- **Recall**: la frazione di soggetti positivi identificati correttamente dal modello rispetto al totale dei positivi reali;
- **F1-score**: la media armonica tra precision e recall, particolarmente utile in presenza di classi sbilanciate, poiché penalizza fortemente la presenza di numerosi falsi positivi (FP) o falsi negativi (FN);
- **Macro average (macro avg)**: media aritmetica delle metriche calcolate separatamente per ciascuna classe, senza considerare la proporzione delle classi. È utile per valutare il comportamento del modello sulle classi meno rappresentate;

[5.1] Si veda la documentazione ufficiale: `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html`

- **Weighted average (weighted avg)**: media ponderata delle metriche rispetto alla numerosità delle classi. Fornisce una visione globale della performance tenendo conto dello sbilanciamento delle classi nel dataset.

Oltre alle metriche aggregate, vengono esplicitamente considerati anche il numero di **falsi positivi** (FP) e **falsi negativi** (FN):

- **Falsi positivi (FP)**: casi in cui un soggetto affidabile viene erroneamente classificato come "Cattivo creditore".
- **Falsi negativi (FN)**: casi in cui un soggetto a rischio viene erroneamente classificato come "Buon creditore".

Queste misure sono fondamentali in ambito creditizio, dove la gravità dell'errore può variare sensibilmente tra Falsi Positivi e Falsi Negativi[5.2].

Nel contesto sperimentale, tali metriche sono calcolate sia per la pipeline agent-based (confronto tra predizione LLM e modello ML), sia rispetto alla *ground truth* di riferimento.

5.2.2 Metriche di interpretabilità e consistenza

Oltre alle metriche di performance predittiva, è stata valutata anche la *consistenza esplicativa* delle decisioni. In particolare, per ogni caso che viene valutato, viene confrontata la motivazione fornita dall'agente LLM (sotto forma di punti di forza e di debolezza) con le feature più influenti individuate dal modello di explainability (SHAP). Questa analisi, realizzata tramite l'interfaccia `Streamlit`, consente di individuare:

- i casi in cui le ragioni espresse dal LLM risultano coerenti con la spiegazione numerica (SHAP), rafforzando la trasparenza e l'affidabilità del sistema;
- i casi di discordanza, utili per individuare limiti o potenziali bias nella conoscenza delle regole e nelle motivazioni apprese dall'LLM.

L'analisi della coerenza tra motivazione e spiegazione viene effettuata a livello qualitativo tramite la demo interattiva, mentre la quantificazione aggregata di tali risultati è oggetto di discussione nelle sezioni successive.

5.3 Confronto tra pipeline classica e architettura agent-based

La valutazione sperimentale della pipeline proposta prevede un'analisi comparativa tra l'approccio tradizionale basato esclusivamente sul modello di machine learning

[5.2] In particolare, i falsi negativi possono comportare rischi economici significativi per l'istituto finanziario, mentre i falsi positivi possono generare perdita di opportunità e insoddisfazione nel cliente.

supervisionato LightGBM e l'architettura agent-based che integra un agente predittore e un agente validatore. In questa sezione, vengono presentati e discussi i risultati ottenuti su vari livelli di confronto, al fine di evidenziare i punti di forza e le eventuali criticità introdotte dall'approccio agent-based.

Le analisi sono basate sui report di classificazione e sulle principali metriche introdotte nella sezione precedente; la discussione è arricchita da considerazioni qualitative sui casi di errore più significativi.

5.3.1 Performance del modello ML (baseline e varianti)

La valutazione della pipeline parte dall'analisi delle prestazioni del modello di machine learning LightGBM, utilizzato come baseline per il confronto con l'architettura agent-based. Nella Tabella 5.1 è riportato il *classification report* relativo al confronto tra le predizioni del modello ML e la ground truth binaria del dataset di riferimento[5.3].

Tabella 5.1: Classification report del modello ML rispetto alla ground truth.

	Precision	**Recall**	**F1-score**	**Support**
Buon creditore	0.9621	0.9854	0.9736	270898
Cattivo creditore	0.1620	0.0676	0.0954	11276
Accuracy				0.9488
Macro avg	0.5620	0.5265	0.5345	282174
Weighted avg	0.9301	0.9488	0.9385	282174

Nel complesso, il modello ML ottiene un'accuratezza globale elevata (94,9%), risultato principalmente dovuto all'elevata capacità di individuare correttamente i *buoni creditori*, che costituiscono la classe largamente predominante nel dataset. In particolare, sia la precision che il recall per questa classe risultano superiori al 96%. Tuttavia, le prestazioni sulla classe minoritaria dei *cattivi creditori* sono nettamente inferiori: il recall scende sotto il 7% e il valore di F1-score rimane al di sotto del 10%. Questo si traduce in un numero considerevole di falsi negativi (10.514), ossia casi di soggetti rischiosi che vengono erroneamente considerati affidabili, e in una quota comunque non trascurabile di falsi positivi (3.942).

Tali risultati mettono in luce i limiti di un modello supervisionato su un problema di classificazione binaria caratterizzato da un marcato sbilanciamento delle classi. La quasi totalità dei clienti viene identificata come "Buon creditore", mentre solo una ristretta minoranza viene riconosciuta come "Cattivo creditore". Inoltre, la natura binaria e conservativa delle etichette di riferimento amplifica la difficoltà di individuare

[5.3] La ground truth del dataset originale [15] attribuisce l'etichetta 1 esclusivamente ai casi a massimo rischio di default, generando una marcata asimmetria tra le due classi.

correttamente i casi realmente rischiosi. A questo si aggiunge il disallineamento tra l'output del modello, che produce valori di rischio nell'intervallo continuo tra 0 e 1, e la natura strettamente binaria del target. Di conseguenza, anche adottando una soglia di classificazione ottimale, si registrano numerosi casi di errore, in particolare nella rilevazione dei soggetti a maggior rischio.

Per provare a superare queste difficoltà sono stati condotti ulteriori esperimenti, introducendo due strategie di riequilibrio delle classi:

- **undersampling 1:1**, in cui il modello è stato addestrato su un campione bilanciato di pari numero di *buoni* e *cattivi creditori*;
- **data augmentation (SMOTE-like**[5.4] **)**, in cui sono stati generati campioni sintetici positivi per riequilibrare le classi mantenendo la numerosità complessiva elevata.

Tabella 5.2: Classification report con addestramento su dati bilanciati.

	Precision	**Recall**	**F1-score**	**Support**
Buon creditore	0.9617	0.9887	0.9750	270898
Cattivo creditore	0.1628	0.0529	0.0798	11276
Accuracy				0.9513
Macro avg	0.5622	0.5208	0.5274	282174
Weighted avg	0.9297	0.9513	0.9392	282174

Con dati bilanciati tramite undersampling (Tabella 5.2), l'accuratezza resta pressoché invariata (95,1%), ma il recall dei cattivi creditori scende ulteriormente (5,3%), confermando le difficoltà del modello nel riconoscere correttamente la classe minoritaria.

Tabella 5.3: Classification report con dati bilanciati tramite data augmentation.

	Precision	**Recall**	**F1-score**	**Support**
Buon creditore	0.9622	0.9807	0.9714	270898
Cattivo creditore	0.1397	0.0755	0.0980	11276
Accuracy				0.9445
Macro avg	0.5510	0.5281	0.5347	282174
Weighted avg	0.9294	0.9445	0.9365	282174

L'uso di un approccio di *data augmentation* (SMOTE numerico, Tabella 5.3) ha permesso un lievissimo miglioramento sul recall della classe minoritaria (7,5%) e sul suo F1-score (9,8%), pur a fronte di una minima riduzione dell'accuratezza complessiva (94,4%).

[5.4] SMOTE (Synthetic Minority Over-sampling Technique) è una tecnica di *oversampling* che genera campioni sintetici per la classe minoritaria interpolando i vettori esistenti nello spazio delle feature, anziché duplicarli [7].

Questi tentativi di riequilibrio tramite undersampling e augmentation non hanno apportato un miglioramento sostanziale delle prestazioni complessive. Pertanto, nelle fasi successive della pipeline è stato adottato il modello **baseline** addestrato sull'intero dataset sbilanciato, considerato comunque più stabile ed efficace rispetto alle varianti.

5.3.2 Performance della pipeline agent-based

L'analisi sperimentale prosegue valutando le prestazioni della pipeline agent-based, in cui la decisione finale è affidata all'agente predittore supportato dalle regole generate e validate a partire dal modello ML di base. In questa sezione, la qualità delle predizioni viene esaminata in alcuni scenari distinti: il primo considera esclusivamente i dati per cui il modello ML fornisce una classificazione corretta, il secondo considera esclusivamente i dati per cui il modello ML fornisce una classificazione errata, mentre il terzo prende in esame l'intero dataset di test. In tutti i casi, per motivi di costi, si considera un campione di mille elementi del dataset preso bilanciando le due classi per garantire una copertura adeguata dei dati.

Questa analisi consente di verificare la capacità dell'agente predittore di generalizzare e interpretare correttamente i dati sia in condizioni ideali, che sfavorevoli, che in un contesto reale, caratterizzato da maggiore eterogeneità e complessità.

Valutazione LLM su campioni con predizione corretta ML

La prima analisi è stata condotta sui campioni del dataset per i quali il modello ML di base aveva già prodotto una classificazione corretta rispetto alla ground truth. In questo scenario non ambiguo, si intende valutare la capacità dell'agente predittore LLM di mantenere la coerenza e la qualità delle predizioni, confermando la robustezza della pipeline agent-based nei casi in cui il sistema tradizionale risulta affidabile.

Come riportato nel classification report in Tabella 5.4, l'agente LLM mostra ottime prestazioni su questo sottoinsieme di dati:

- l'**accuracy** complessiva si attesta all'83,5%;
- **F1-score** globale raggiunge il valore di 0.8374;
- sia la precision che il recall si mantengono superiori all'82% per entrambe le classi.

Tabella 5.4: Classification report dell'agente LLM sui campioni in cui il modello ML fornisce predizione corretta.

	Precision	**Recall**	**F1-score**	**Support**
Buon creditore	0.8233	0.8480	0.8355	500
Cattivo creditore	0.8433	0.8180	0.8305	500
Accuracy				0.8330
Macro avg	0.8333	0.8330	0.8330	1000
Weighted avg	0.8333	0.8330	0.8330	1000

Questi risultati evidenziano la capacità dell'agente LLM di replicare efficacemente le decisioni del modello ML nei casi non ambigui, senza introdurre errori sistematici o peggioramenti delle performance. In particolare, il buon equilibrio tra precisione e recall nelle due classi conferma che la pipeline agent-based è in grado di mantenere elevata affidabilità anche su dati non esplicitamente visti in fase di addestramento o validazione delle regole, preservando la robustezza della soluzione complessiva. Il basso numero di errori, equamente suddivisi tra falsi positivi e falsi negativi, testimonia inoltre la stabilità del sistema e la sua capacità di adattarsi efficacemente ai pattern già appresi dal modello di base, minimizzando il rischio di degradazione della qualità predittiva. È importante sottolineare che, in quanto sono considerati solo i casi in cui la predizione del modello ML coincide con la ground truth, le metriche ottenute dall'agente LLM riflettono una situazione favorevole, in cui le condizioni di input sono prive di ambiguità e di casi limite. Questo spiega le ottime performance osservate, confermando che la pipeline agent-based è in grado di mantenere coerenza e robustezza nei contesti in cui il sistema tradizionale fornisce già predizioni affidabili.

Valutazione LLM su campioni con predizione errata ML

Questa analisi si concentra esclusivamente sui casi in cui il modello ML fornisce una predizione errata rispetto alla ground truth. L'obiettivo è verificare come si comporta l'agente LLM in queste situazioni ambigue e se sia in grado di correggere, o al contrario replicare, l'errore commesso dal modello ML.

Come riportato nella Tabella 5.5, il confronto tra LLM e ML evidenzia una buona coerenza tra i due sistemi anche in questo scenario:

- l'**accuracy** si attesta al 77,9%;
- **F1-score** raggiunge il valore di 0.7788;
- sia precisione che recall risultano ben bilanciate su entrambe le classi, attestandosi intorno al 77%-78%.

Tabella 5.5: Classification report LLM vs ML su campioni dove ML ha predetto erroneamente.

	Precision	**Recall**	**F1-score**	**Support**
Buon creditore	0.7784	0.7800	0.7792	500
Cattivo creditore	0.7796	0.7780	0.7788	500
Accuracy				0.7790
Macro avg	0.7790	0.7790	0.7790	1000
Weighted avg	0.7790	0.7790	0.7790	1000

Tuttavia, se si esamina la capacità dell'agente LLM di correggere gli errori del modello ML rispetto alla ground truth, emerge un quadro differente (Tabella 5.6):

- l'**accuracy** rispetto alla ground truth scende al 22,1
- **F1-score** cala drasticamente a 0.2202;
- si osserva un sostanziale equilibrio tra falsi positivi e falsi negativi, ciascuno prossimo a 390 su 500.

Tabella 5.6: Classification report LLM vs ground truth su campioni dove il modello ML ha predetto erroneamente.

	Precision	**Recall**	**F1-score**	**Support**
Buon creditore	0.2216	0.2220	0.2218	500
Cattivo creditore	0.2204	0.2200	0.2202	500
Accuracy				0.2210
Macro avg	0.2210	0.2210	0.2210	1000
Weighted avg	0.2210	0.2210	0.2210	1000

Questi risultati mostrano chiaramente che, nei casi in cui il modello ML si discosta dalla ground truth, l'agente LLM tende a replicare le scelte del modello ML, mantenendo alta la coerenza interna della pipeline agent-based. Tuttavia, ciò comporta una perdita significativa di accuratezza rispetto alla realtà, evidenziando che il LLM, in quanto validato e aggiornato principalmente sui dati forniti dal modello ML, ne eredita inevitabilmente anche le limitazioni e gli errori sistematici. Questa analisi conferma che il comportamento dell'agente LLM è fortemente guidato dalle regole apprese sulla base delle predizioni del modello ML sottostante. Sebbene ciò garantisca stabilità e coerenza nel sistema, sottolinea anche la necessità di strategie di validazione più articolate, eventualmente basate su supervisione diretta rispetto alla ground truth, per aumentare la capacità della pipeline di correggere autonomamente gli errori del modello tradizionale.

Valutazione LLM su campioni generici

Questa analisi prende in esame le prestazioni dell'agente LLM su un campione generico bilanciato del dataset di test, senza alcuna selezione preventiva. Tale scenario, più vicino alle condizioni operative reali, permette di valutare la capacità della pipeline agent-based di mantenere performance robuste su dati eterogenei e complessi, e di confrontarla sia con il modello ML che rispetto alla ground truth.

Il classification report in Tabella 5.7 mostra che l'agente LLM mantiene un'elevata coerenza rispetto al modello ML di riferimento:

- **Accuracy** LLM vs ML: 82,8%;
- **F1-score** (Cattivo creditore): 0.8227;
- Precision e recall bilanciati tra le classi, con valori superiori all'80%.

Tabella 5.7: Classification report dell'agente LLM vs ML su campioni generici del dataset.

	Precision	**Recall**	**F1-score**	**Support**
Buon creditore	0.8094	0.8580	0.8330	500
Cattivo creditore	0.8489	0.7980	0.8227	500
Accuracy				0.8280
Macro avg	0.8292	0.8280	0.8278	1000
Weighted avg	0.8292	0.8280	0.8278	1000

Tuttavia, il confronto rispetto alla ground truth evidenzia che sia il modello ML che la pipeline LLM, pur mantenendo un buon livello di accuratezza sui "Buoni creditori", faticano nell'identificare correttamente i "Cattivi creditori". È importante sottolineare che in questo scenario il dataset presenta una marcata asimmetria tra le due classi: i "Buoni creditori" costituiscono la grande maggioranza dei campioni (899 su 1000), mentre i "Cattivi creditori" rappresentano soltanto 101 casi, poichè il campionamento è stato effettuato sull'etichetta *rule_brcg* e non sull'etichetta *target*, ovvero il reale ground truth. La Tabella 5.8 riporta in dettaglio le metriche di confronto tra modello ML e pipeline LLM rispetto alla ground truth su un campione generico.

- **ML vs GT**: l'accuracy si attesta al 56,3%, con un F1-score di appena 0.2729 per la classe minoritaria, a fronte di una forte asimmetria negli errori: il modello tende a essere molto conservativo, classificando la maggior parte dei casi come affidabili, e producendo così un numero elevato di falsi positivi (418) rispetto ai falsi negativi (19).

- **LLM vs GT**: l'accuracy raggiunge il 58,1%, con un F1-score per i "Cattivi creditori" di 0.2662. Rispetto al modello ML, si osserva una ridistribuzione degli errori, con una leggera diminuzione dei falsi positivi (394) e un lieve incremento dei falsi negativi (25). Questo risultato indica che l'agente LLM, pur mantenendo una certa coerenza con il modello ML sottostante, replica i pattern di errore indotti dall'asimmetria delle classi, mostrando una limitata capacità di correggere le mancate identificazioni dei casi realmente rischiosi.

Tabella 5.8: Confronto tra ML e LLM rispetto alla ground truth su campioni generici (support: 899 Buoni, 101 Cattivi).

Scenario	**Accuracy**	**F1-score** (Cattivo)	**FP**	**FN**
ML vs GT	0.5630	0.2729	418	19
LLM vs GT	0.5810	0.2662	394	25

L'agente LLM conferma la capacità della pipeline agent-based di mantenere una forte coerenza interna rispetto al modello ML su cui è stato validato, garantendo performance elevate in termini di accordo tra le due classi. Tuttavia, la valutazione rispetto alla ground truth evidenzia come entrambe le soluzioni siano fortemente condizionate dalla natura sbilanciata del dataset, che penalizza la rilevazione efficace dei "Cattivi creditori". Questi risultati rafforzano la necessità di arricchire ulteriormente la base di conoscenza e i criteri di validazione, al fine di ridurre la propagazione degli errori sistematici del modello tradizionale e aumentare la capacità del sistema di intercettare correttamente i casi realmente rischiosi anche in presenza di forte squilibrio tra le classi.

5.3.3 Discussione e sintesi comparativa

Alla luce delle analisi svolte, è possibile tracciare un quadro comparativo tra il modello ML tradizionale (LightGBM) e la pipeline agent-based con LLM. È importante precisare che, mentre i risultati del modello ML (baseline) sono calcolati sull'intero dataset di test, le valutazioni relative all'agente LLM sono state condotte su campioni bilanciati di dimensione ridotta, scelti per ragioni di costo computazionale, ma garantendo rappresentatività di entrambe le classi.

La Tabella 5.9 riassume le principali metriche nei diversi scenari considerati.

Tabella 5.9: Confronto tra modello ML (LightGBM) e pipeline agent-based (LLM) nei diversi scenari di test.

Scenario	Accuracy	F1-score	FP	FN
ML vs Ground Truth (intero dataset)	0.9488	0.0954	3,942	10,514
LLM su dati con ML corretto	0.8350	0.8374	90	75
LLM su dati con ML errato	0.7790	0.7788	110	111
LLM su campioni generici	0.8280	0.8227	71	102
ML vs GT su campioni generici	0.5630	0.2729	418	19
LLM vs GT su campioni generici	0.5810	0.2662	394	25

L'F1-score della baseline ML fa riferimento alla classe minoritaria ("Cattivo creditore") sull'intero dataset.

Dall'analisi emergono alcune informazioni rilevanti:

- **Modello ML**: ottiene un'accuratezza molto elevata sull'intero dataset (oltre il 94%), ma questa performance è dovuta principalmente alla capacità di riconoscere la classe maggioritaria ("Buoni creditori"). Le metriche sulle classi minoritarie sono critiche: l'F1-score sui "Cattivi creditori" resta inferiore al 10%, con un numero molto elevato di falsi negativi, ovvero soggetti rischiosi non intercettati dal sistema.

- **Pipeline agent-based**: Sui casi "facili" (dove ML è corretto), l'agente LLM mantiene performance elevate (accuracy e F1-score > 83%), mostrando robustezza nel replicare decisioni affidabili. Nei casi più sfidanti (dove ML sbaglia), l'LLM tende a replicare i pattern del modello ML di riferimento, senza mostrare una significativa capacità di correzione autonoma: accuracy e F1-score, in generale, si mantengono comunque superiori al 77%. Su campioni generici bilanciati, la pipeline agent-based continua a mostrare equilibrio tra le classi, mantenendo F1-score e accuracy oltre l'82%.

- **Confronto rispetto alla ground truth su campioni generici**: Sia ML che LLM, se valutati rispetto alle etichette reali, mostrano un deciso calo di accuratezza e F1-score (entrambe le soluzioni si attestano tra il 56% e il 58% di accuracy e intorno allo 0.27 di F1-score per i "Cattivi creditori"). È importante notare che in questi test la numerosità della classe "Cattivo creditore" è molto inferiore (101 su 1000) rispetto ai "Buoni creditori" (899 su 1000), il che rende ancora più critico il compito di rilevazione dei soggetti a rischio e penalizza fortemente le metriche aggregate. LLM e ML producono errori simili (falsi positivi e falsi negativi), a dimostrazione di come l'LLM, basato sulle regole validate a partire da ML, ne erediti anche le limitazioni strutturali.

Limiti e considerazioni applicative: Le metriche dell'agente LLM si basano su campioni ridotti e bilanciati rispetto all'intero dataset, mentre la baseline ML riflette il

comportamento su tutta la popolazione. Pertanto, le conclusioni sulla generalizzazione andrebbero confermate su dati più estesi e in scenari operativi reali, specie in presenza di forti squilibri tra le classi. Inoltre, la dipendenza dell'agente LLM dalle regole ML limita la sua capacità di correggere autonomamente gli errori del modello tradizionale.

Potenzialità e casi d'uso: L'architettura agent-based si conferma robusta e affidabile su dati bilanciati e casi privi di ambiguità, ed è in grado di mantenere performance competitive anche su dati eterogenei. Il vero valore aggiunto emerge nella trasparenza, nell'aggiornabilità delle regole e nella capacità di tracciare le motivazioni delle decisioni, aspetti cruciali in applicazioni bancarie, dove la riduzione dei rischi e la spiegabilità sono priorità assolute. L'integrazione futura di strategie di validazione diretta sulla ground truth o di arricchimento della base di conoscenza potrà ulteriormente incrementare la capacità del sistema di intercettare soggetti a rischio, riducendo gli errori ereditati dal modello ML.

5.4 Explainability e aggiornamento delle regole

All'interno della pipeline agent-based, le tecniche di explainability - in particolare l'utilizzo di SHAP e l'analisi delle motivazioni fornite dall'agente LLM - giocano un ruolo centrale nel processo di aggiornamento dinamico della knowledge base. Di seguito si presenta un esempio concreto che illustra come l'architettura proposta consenta di integrare spiegazioni locali e regole aggiornabili, con l'obiettivo di aumentare coerenza, trasparenza e capacità predittiva del sistema.

5.4.1 Esempio: aggiornamento delle regole per il cliente indice 4

Nel caso del cliente identificato dall'indice 4 (`case_id` 39251), il sistema ha valutato un profilo caratterizzato da valori che hanno portato alla classificazione da parte del modello LLM come *Buon creditore*, mentre il modello ML lo ha individuato come *Cattivo creditore*. La discordanza tra i due sistemi ha quindi attivato il meccanismo di aggiornamento delle regole.

Motivazioni LLM

Punti di forza individuati:

- `pmtnum_254L`: alto numero di pagamenti (12)
- `mobilephncnt_593L`: bassa condivisione del telefono (1)
- `annuity_780A`: annuity moderata (3676,40)
- `max_empl_employedfrom_271D`: periodo di impiego stabile (-322)

- meanAP_totalamount_6A_9: importo medio impegnato modesto (21.898,6)

Punti di debolezza individuati:

- mean_numberofoverdueinstlmax_1151L_9: valore molto alto (2384)
- min_overdueamountmax_155A_9: scostamento negativo (2049,678)
- meanAP_overdueamountmax_155A_9: importo scostamento importante (2049,678)
- min_overdueamountmax_35A_9: scostamento rilevante (9000)

Nuove regole generate dall'agente validatore

A seguito dell'analisi dei valori SHAP, il sistema ha suggerito l'aggiornamento o la creazione delle seguenti regole:

1. mean_numberofoverdueinstlmax_1151L_9 > 2000 **AND** meanAP_overdueamountmax_15 > 2000
2. max_empl_employedfrom_271D > -300 **AND** mean_numberofoverdueinstlmax_1039L_9 > 30
3. annuity_780A > 3500 **AND** pmtnum_254L < 20

Tali regole vengono integrate nella knowledge base per affinare i criteri decisionali e consentire al sistema di adattarsi in modo dinamico a pattern ricorrenti di rischio, precedentemente sottovalutati.

Valutazione e implicazioni

Questo esempio dimostra come la pipeline agent-based sia in grado di:

- identificare in modo trasparente i principali fattori di rischio;
- confrontare e integrare motivazioni testuali e spiegazioni numeriche;
- aggiornare la knowledge base aggiungendo o modificando regole ogniqualvolta emerga una nuova evidenza;
- migliorare progressivamente la coerenza tra spiegazione, motivazione e azione, incrementando trasparenza e affidabilità del sistema.

Tale meccanismo di aggiornamento dinamico, guidato dalle tecniche di explainability, rappresenta un avanzamento sostanziale rispetto ai tradizionali sistemi black-box, facilitando sia l'adattamento automatico del sistema che il rispetto dei requisiti di tracciabilità e audit richiesti in ambito finanziario.

5.5 Esempio di caso studio

In questa sezione si presenta un esempio concreto, estratto dalla demo interattiva realizzata in Streamlit, che evidenzia sia un caso di discordanza tra la predizione dell'agente LLM e il modello ML (cliente indice 5), sia un caso concordanza (cliente indice 0). Tali esempi mostrano in modo trasparente come la pipeline agent-based consenta non solo di motivare ogni decisione, ma anche di intervenire in modo coerente e interpretabile nel processo di aggiornamento delle regole, nonostante la demo interattiva sia sviluppata solo a scopo di visualizzazione delle predizione.

5.5.1 Caso 1: Discordanza tra LLM e modello ML (Cliente #5)

Nella Figura 5.1 è illustrato il risultato relativo al cliente con indice esterno 5. In questo caso, si osserva una discordanza tra le valutazioni:

- L'agente LLM predice **Cattivo creditore**, motivando la scelta sulla base di diversi punti di debolezza, come l'alto numero di richieste recenti e il valore elevato della rata mensile (`annuity_780A`).
- Il modello di ML, al contrario, classifica il cliente come **Buon creditore** con probabilità di default pari a 68.6%, in accordo con la ground truth.

L'analisi di explainability mostra la corrispondenza (o discordanza) tra le feature principali individuate da SHAP e le motivazioni testuali fornite dall'LLM. Nella tabella riportata a sinistra viene evidenziato come, per alcune feature chiave (ad esempio `max_empl_employedfrom_271D` e `numreqtestsn_859L`), le motivazioni LLM risultino incoerenti rispetto alla direzione dell'importanza SHAP, mostrando quali valori sarebbero stati utilizzati dall'agente validatore per aggiornare le regole.

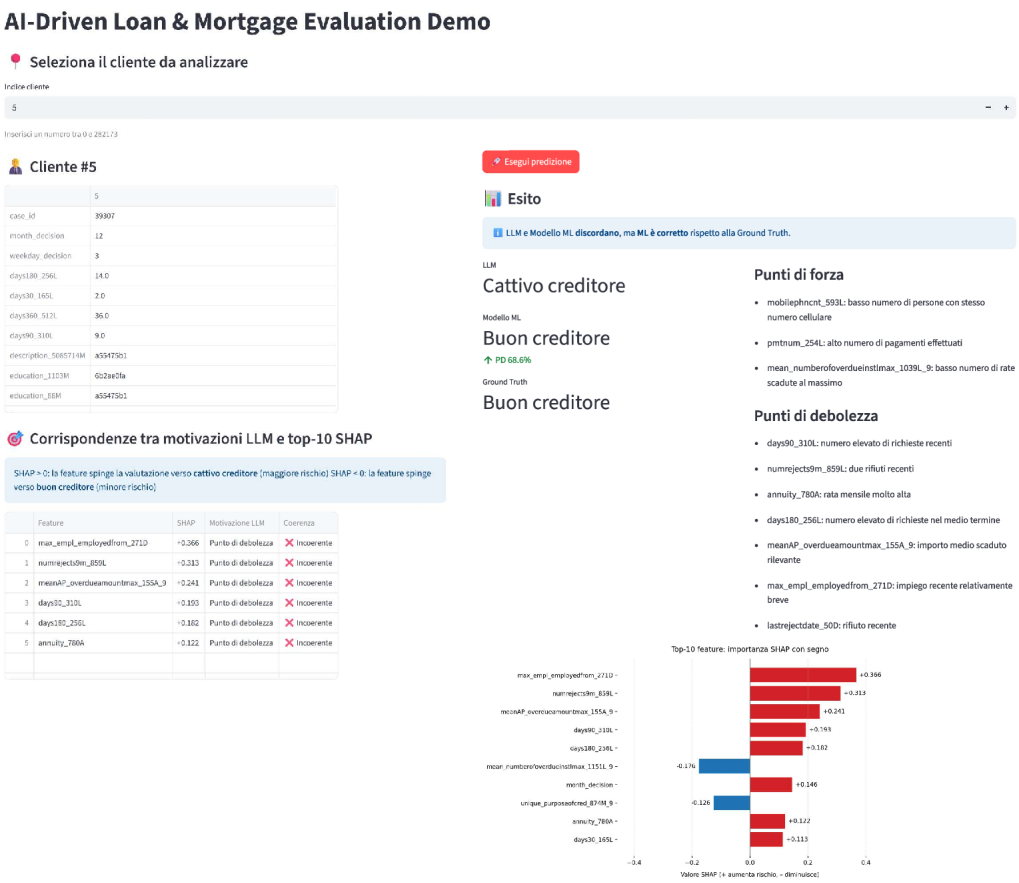

Figura 5.1: Esempio di discordanza tra LLM e modello ML per il cliente 5: analisi dettagliata delle motivazioni, punti di forza/debolezza e corrispondenza con le feature SHAP.

5.5.2 Caso 2: Concordanza tra LLM e modello ML (Cliente #0)

Come termine di confronto, la Figura 5.2 mostra il risultato per il cliente con indice esterno 0, in cui LLM e modello ML forniscono una valutazione perfettamente concorde:

- Sia LLM che ML predicono **Buon creditore**, in accordo con la ground truth.
- I punti di forza e debolezza che vengono evidenziati dall'LLM risultano coerenti sia con le motivazioni numeriche (SHAP), sia con l'esito complessivo.

In questo scenario, la tabella delle corrispondenze mostra comunque alcune differenze puntuali tra il segno dell'importanza SHAP e la categorizzazione come punto di forza/debolezza da parte dell'LLM, ma nel complesso la decisione finale rimane robusta e interpretabile.

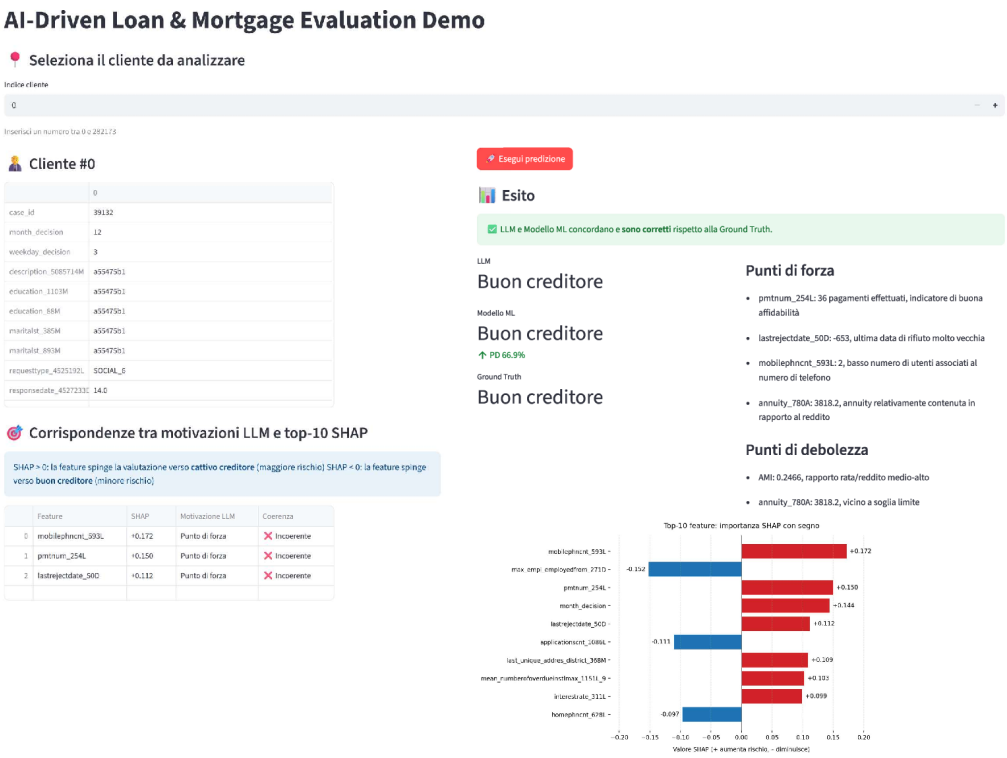

Figura 5.2: Esempio di concordanza tra LLM e modello ML per il cliente 0: esito, motivazioni e confronto tra spiegazione numerica e testuale.

5.5.3 Discussione

Questi esempi dimostrano la trasparenza e il valore aggiunto dell'architettura agent-based:

- Permettono di analizzare, in modo puntuale e visuale, le motivazioni sottostanti alle decisioni dei diversi agenti;
- Offrono uno strumento operativo per individuare rapidamente eventuali incoerenze tra spiegazioni testuali e numeriche;
- Consentono, grazie al processo di validazione e aggiornamento delle regole, di intervenire tempestivamente per migliorare la coerenza e l'affidabilità del sistema decisionale.

La possibilità di visualizzare in un'unica interfaccia tutte le informazioni chiave — predizioni di modello ML e LLM, motivazioni espresse, feature SHAP, punti di forza e debolezza, corrispondenza con la ground truth — si rivela estremamente utile per un'analisi critica e approfondita del comportamento della pipeline. Questo approccio consente di verificare in modo immediato e trasparente se il modello tradizionale e l'agente LLM utilizzano le stesse variabili in modo coerente, individuando tempestivamente eventuali

discrepanze o errori sistematici. Il confronto simultaneo di tutte le componenti rende il sistema non solo interpretabile, ma anche controllabile e migliorabile nel tempo, garantendo un processo decisionale più affidabile, tracciabile e adatto a contesti di alta responsabilità come la valutazione automatica del rischio creditizio.

Conclusioni

Il lavoro svolto in questa tesi ha preso avvio dalla consapevolezza che i processi di valutazione del rischio di credito richiedono oggi non soltanto modelli predittivi accurati, ma anche strumenti in grado di garantire leggibilità, verificabilità e capacità di adattamento. L'obiettivo principale non è stato quindi quello di perseguire unicamente un miglioramento delle metriche di classificazione, ma di costruire un sistema integrato che coniugasse analisi statistica, apprendimento automatico e ragionamento basato su regole, così da supportare decisioni più affidabili e tracciabili.

I risultati ottenuti hanno mostrato come l'approccio agent-based, fondato sull'interazione tra modello di machine learning e modello linguistico, rappresenti una soluzione promettente. Il predittore ha consentito di derivare decisioni coerenti con le regole della knowledge base, mentre il validatore ha svolto un ruolo cruciale nel rilevare eventuali incoerenze e nell'aggiornare dinamicamente le regole stesse, sfruttando le informazioni derivate dall'analisi dei valori SHAP. Questa sinergia ha dimostrato che accuratezza predittiva e trasparenza possono coesistere in un sistema unitario capace di giustificare ogni decisione attraverso motivazioni esplicite e ancorate ai dati.

Un aspetto rilevante emerso dall'analisi effettuata è che le regole derivate dai valori SHAP non costituiscono soltanto un output descrittivo, ma diventano uno strumento di mediazione tra il modello e l'operatore umano: un linguaggio condiviso che permette di connettere vincoli statistici, criteri normativi e decisione finale. In questa prospettiva, la tracciabilità delle decisioni è parte integrante del processo. Inoltre, la possibilità di aggiornare la base di conoscenza si è rivelata fondamentale per garantire le prospettive evolutive del sistema, evitando la cristallizzazione di regole obsolete.

Naturalmente, sono emersi anche dei limiti. La validità del sistema resta fortemente legata alla qualità e alla rappresentatività dei dati di partenza e del modello di machine learning alla base; un aggiornamento troppo frequente della knowledge base comporta il rischio di adattamento al rumore; e le spiegazioni generate, per quanto utili alla comprensione, non devono essere interpretate come prove di causalità, ma come strumenti di supporto decisionale. Anche l'implementazione pratica di un simile approccio comporta un onere operativo aggiuntivo, che deve essere valutato attentamente dalle istituzioni finanziarie interessate ad adottarlo.

Dallo studio si possono trarre alcune linee guida utili. Tra queste, la necessità di

valutare le performance non solo in termini aggregati, ma anche su segmenti omogenei di clientela, così da garantire maggiore equità e robustezza; la rilevanza di un monitoraggio continuo, in grado di intercettare eventuali drift nei dati e nelle regole; e l'importanza di affiancare alle metriche tradizionali indicatori capaci di distinguere l'impatto di falsi positivi e falsi negativi, che nel contesto creditizio hanno conseguenze molto diverse.

Le prospettive future appaiono particolarmente interessanti. Infatti, oltre al consolidamento di un ciclo di vita già strutturato, uno sviluppo di lungo periodo potrebbe essere rappresentato dall'adozione dei modelli linguistici addestrati sui dati interni di ciascun istituto. In tale scenario, gli LLM potrebbero operare come veri e propri assistenti digitali, capaci non soltanto di affiancare le decisioni di credito, ma anche di prevenire frodi, segnalare tempestivamente posizioni a rischio e simulare scenari di portafoglio. Sebbene ciò richieda ulteriori approfondimenti e adeguati strumenti di governance, si tratta di un percorso che delinea prospettive di forte impatto sia scientifico sia applicativo.

Il contributo principale di questa tesi risiede nell'aver mostrato che è possibile conciliare affidabilità delle previsioni e chiarezza del processo decisionale attraverso un sistema che integra modelli di machine learning e modelli linguistici in un flusso unitario. Tale risultato non è banale, poiché spesso le soluzioni tradizionali nel campo del credit scoring si concentrano esclusivamente sulla massimizzazione delle prestazioni quantitative, sacrificando la leggibilità del processo decisionale e rendendo difficile giustificare le scelte effettuate. Al contrario, il sistema proposto dimostra che un approccio ibrido consente di preservare l'accuratezza e di restituire motivazioni esplicite e comprensibili, ancorate ai dati effettivamente rilevanti per ciascun cliente.

Questo aspetto rappresenta un elemento di novità rispetto ai modelli black-box: l'adozione di regole dinamiche, derivate sia da metriche di explainability sia dalla supervisione dell'agente validatore, consente infatti di rendere tracciabile ogni passaggio e di collegare la previsione numerica con una giustificazione testuale coerente. Ciò apre la strada a pratiche decisionali più difendibili dal punto di vista normativo, oltre che più comprensibili per gli operatori del settore e, potenzialmente, anche per i clienti stessi.

Bibliografia

[1] Matteo Altieri. «An Innovative Approach to Scorecard Models». Tesi Magistrale in Data Science, sviluppata presso DATA Reply. Tesi di laurea mag. Università degli Studi di Milano-Bicocca, 2024.

[2] Vijay Arya et al. «AI Explainability 360: Impact and Design». In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 11. IBM Research. 2022, pp. 12014–12022. URL: https://arxiv.org/pdf/2109.12151.

[3] Banca d'Italia. *Disposizioni di Vigilanza per le banche – Circolare n. 285 del 17 dicembre 2013, aggiornata alla versione n. 49 del 23 luglio 2024*. 2024. URL: https://www.bancaditalia.it/compiti/vigilanza/normativa/archivio-norme/circolari/c285.

[4] Banca d'Italia. *Nota n. 13 del 20 luglio 2021 – Attuazione degli Orientamenti dell'Autorità bancaria europea in materia di concessione e monitoraggio dei prestiti (EBA/GL/2020/06)*. 2021. URL: https://www.bancaditalia.it/compiti/vigilanza/normativa/orientamenti-vigilanza/elenco-esa/note/Nota-n-13-del-20-luglio-2021.pdf.

[5] Deepanshu Bhalla. *Weight of Evidence (WOE) and Information Value (IV) Explained*. Consultato il 13 luglio 2025. 2015. URL: https://www.listendata.com/2015/03/weight-of-evidence-woe-and-information.html.

[6] Leo Breiman. «Random Forests». In: *Machine Learning* 45.1 (2001), pp. 5–32. URL: https://doi.org/10.1023/A:1010933404324.

[7] N. V. Chawla, K. W. Bowyerand L. O. Hall e W. P. Kegelmeyer. «SMOTE: Synthetic Minority Over-sampling Technique». In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357. URL: https://doi.org/10.48550/arXiv.1106.1813.

[8] Tianqi Chen e Carlos Guestrin. «XGBoost: A scalable tree boosting system». In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 785–794. URL: https://doi.org/10.1145/2939672.2939785.

[9] J. N. Crook, D. B. Edelman e L. C. Thomas. «Recent developments in consumer credit risk assessment». In: *European Journal of Operational Research* 183.3 (2007), pp. 1447–1465. URL: https://doi.org/10.1016/j.ejor.2006.09.100.

[10] S. Dash, O. Günlük e D. Wei. «Boolean Decision Rules via Column Generation». In: *arXiv preprint arXiv:1805.09901* (2020). URL: https://doi.org/10.48550/arXiv.1805.09901.

[11] Tonmoy Debnath et al. «A Comprehensive Survey of Prompt Engineering Techniques in Large Language Models». In: *TechRxiv* (2025). URL: https://digitalcommons.odu.edu/ece_fac_pubs/514.

[12] European Banking Authority (EBA). *Guidelines on loan origination and monitoring*. 2020. URL: https://www.eba.europa.eu/activities/single-rulebook/regulatory-activities/credit-risk/guidelines-loan-origination-and-monitoring.

[13] Shanshan Han et al. «LLM Multi-Agent Systems: Challenges and Open Problems». In: *arXiv preprint arXiv:2402.03578* (2024). URL: https://doi.org/10.48550/arXiv.2402.03578.

[14] David J. Hand e W. E. Henley. «Statistical classification methods in consumer credit scoring: a review». In: *Journal of the Royal Statistical Society: Series A* 160.3 (1997), pp. 523–541. URL: https://doi.org/10.1111/j.1467-985X.1997.00078.x.

[15] *Home Credit - Credit Risk Model Stability*. https://www.kaggle.com/competitions/home-credit-credit-risk-model-stability/data. Dataset pubblicato su Kaggle. 2022.

[16] Tu D. Huynh et al. «Explainability-by-Design: A Methodology to Support Explanations in Decision-Making Systems». In: *arXiv preprint arXiv:2206.06251* (2022). URL: https://arxiv.org/pdf/2206.06251v1.

[17] Guolin Ke, Qi Meng, Thomas Finley et al. «LightGBM: A Highly Efficient Gradient Boosting Decision Tree». In: *Advances in Neural Information Processing Systems* (2017). URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2017/11/lightgbm.pdf.

[18] Sven Lessmann et al. «Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research». In: *European Journal of Operational Research* 247.1 (2015), pp. 124–136. URL: https://doi.org/10.1016/j.ejor.2015.05.030.

[19] Scott M. Lundberg e Su-In Lee. «A Unified Approach to Interpreting Model Predictions». In: *Advances in Neural Information Processing Systems* 30 (2017), pp. 4765–4774. URL: https://doi.org/10.48550/arXiv.1705.07874.

[20] Neel Nanda et al. «On the Biology of a Large Language Model». In: (2025). Sezione: Multilingual circuit. URL: https://transformer-circuits.pub/2025/attribution-graphs/biology.html#dives-multilingual.

[21] *Regolamento (UE) 2024/1689 del Parlamento Europeo e del Consiglio del 13 giugno 2024 sull'Intelligenza Artificiale (AI Act)*. 2024. URL: https://eur-lex.europa.eu/legal-content/IT/TXT/PDF/?uri=OJ:L_202401689.

[22] Laria Reynolds e Kyle McDonell. «Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm». In: *arXiv preprint arXiv:2102.07350* (2021). URL: https://doi.org/10.48550/arXiv.2102.07350.

[23] Marco Tulio Ribeiro, Sameer Singh e Carlos Guestrin. «Why Should I Trust You?: Explaining the Predictions of Any Classifier». In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 1135–1144. URL: https://doi.org/10.1145/2939672.2939778.

[24] Pranab Sahoo et al. «A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications». In: *arXiv preprint arXiv:2402.07927* (2024). URL: https://doi.org/10.48550/arXiv.2402.07927.

[25] S. Yalnizyan. *Intro to Credit Scorecard*. Consultato il 13 luglio 2025. 2018. URL: https://medium.com/data-science/intro-to-credit-scorecard-9afeaaa3725f.

[26] I-Cheng Yeh e Che-hui Lien. «The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients». In: *Expert Systems with Applications* 36.2 (2009), pp. 2473–2480. URL: https://doi.org/10.1016/j.eswa.2007.12.020.