



Portfolio Project — Data Science

Versione 1.0 | February, 2026

MODELLO DI CREDIT SCORING CON MACHINE LEARNING SU DATI BANCARI

Policy decisionale cost-sensitive con vincolo di approvazione, interpretabilità e sensitivity analysis

Autore: Lorenzo Scardilli

Documento tecnico. Salvo diversa indicazione, il testo e le visualizzazioni originali di questo report sono protetti da diritto d'autore; è consentita la citazione con attribuzione. I dataset sono utilizzati esclusivamente a fini di studio e dimostrazione.

Sintesi esecutiva

Scopo. Costruire una **policy operativa** di approvazione/rifiuto del credito che trasformi la probabilità di default (*BAD*) in una decisione, ottimizzando un trade-off realistico tra **rischio, volumi e costo atteso degli errori**.

Impostazione decisionale. Il problema è formulato in ottica *cost-sensitive*: l'errore più critico è approvare un cliente *BAD* (falso negativo), penalizzato più del rifiuto di un cliente *GOOD* (falso positivo). La soglia t è stimata **solo su training** via cross-validation minimizzando $FP + 5 \cdot FN$ e imponendo un vincolo operativo **approval** $\geq 50\%$; l'holdout test è usato una sola volta come verifica finale.

Risultato. La migliore soluzione operativa è **LogReg_NO_SENS** con $t = 0.25$: a parità di vincolo sui volumi, ottiene il **costo minimo** e mantiene una **qualità del portafoglio approvato** più alta rispetto alle alternative considerate. La scelta di Logistic Regression è motivata anche da **interpretabilità** (driver leggibili via *odds ratio*) e maggiore controllabilità in ottica di governance.

Robustezza. Una sensitivity analysis mostra che, nel regime operativo rilevante ($approval \geq 50\%$), la soglia ottimale resta stabile intorno a $t \approx 0.25$ per un intervallo plausibile di asimmetria dei costi ($c_{FN} : c_{FP}$).

Table 1: KPI principali su holdout test (modello finale).

Costo totale ($FP + 5 \cdot FN$)	96
Costo per osservazione	0.48
Approval rate (pred <i>GOOD</i>)	0.52
Bad rate tra approvati	0.096
Recall <i>BAD</i>	0.833
Precision <i>BAD</i>	0.521
ROC-AUC	0.803
PR-AUC	0.651

Nota: il costo è espresso in unità relative (scenario) e serve a confrontare policy sotto la stessa matrice dei costi; non rappresenta una stima monetaria.

Contents

Sintesi esecutiva	1
1 Introduzione	3
1.1 Contesto e obiettivo	3
1.2 Impostazione generale del lavoro	3
2 Dataset	5
3 Metodologia	6
3.1 Setup sperimentale e pipeline	6
3.2 Regola decisionale e funzione di costo	6
3.3 Confronto modelli: criterio operativo e governance	6
4 Risultati	8
4.1 Confronto tra modelli (holdout test)	8
4.2 Performance operativa del modello finale	9
4.3 Scelta della soglia: evidenza del trade-off costo–volume	10
4.4 Interpretabilità: cosa guida il rischio (driver)	11
4.5 Robustezza: sensitivity su costi e vincoli	13
5 Conclusioni	15
5.1 Sintesi dei risultati	15
5.2 Raccomandazioni operative	15
6 Limiti e assunzioni	16
A Note tecniche: metriche e frammenti di codice essenziali	17
A.1 Metriche (definizioni sintetiche)	17
A.2 Codice essenziale (riproducibilità delle scelte chiave)	17
A.2.1 Funzione di costo e ricerca della soglia con vincolo di approval .	17
A.2.2 Cross-validation: soglia stimata sul training (anti-leakage)	18
A.2.3 Driver del rischio: coefficienti e odds ratio (Logistic Regression) .	18

1 Introduzione

1.1 Contesto e obiettivo

Il *credit scoring* è un problema tipico di decisione data-driven: non basta stimare un rischio, occorre trasformare tale stima in una scelta operativa (approvare o rifiutare una richiesta di credito) che sia coerente con obiettivi economici e vincoli di business. In questo progetto si sviluppa una pipeline di Machine Learning per stimare la probabilità che una richiesta appartenga alla classe *BAD* e, soprattutto, per definire una **policy decisionale** applicabile in pratica.

L'output atteso non è quindi solo un modello predittivo “con buone metriche”, ma una regola di decisione esplicita e replicabile, in grado di bilanciare:

- **volume di erogazione** (quante richieste vengono approvate);
- **qualità del portafoglio approvato** (quanti *BAD* passano tra gli approvati);
- **costo atteso degli errori**, distinguendo tra errori con impatti molto diversi.

In ambito credito gli errori non sono equivalenti. Rifiutare un cliente affidabile (*falso positivo*) comporta un costo opportunità (margini perso, cliente perso), mentre approvare un cliente rischioso (*falso negativo*) è più critico perché può tradursi in perdite attese e costi di recupero. Per questo motivo l'ottimizzazione è stata impostata in ottica **cost-sensitive**, utilizzando una funzione di costo asimmetrica:

$$\text{Costo}(t) = 1 \cdot FP(t) + 5 \cdot FN(t),$$

dove *FP* rappresenta *GOOD* rifiutati e *FN* rappresenta *BAD* approvati.

La scelta della soglia t è centrale: determina simultaneamente volumi approvati e rischio residuo nel portafoglio. Per evitare soluzioni artificialmente “conservative” (che minimizzano il costo solo perché rifiutano quasi tutto), è stato imposto un vincolo operativo di *approval rate* minimo:

$$\text{approval}(t) \geq 0.50.$$

La soglia è stimata esclusivamente sul training set tramite cross-validation; l'holdout test viene utilizzato una sola volta per la valutazione finale, riducendo il rischio di *data leakage*.

1.2 Impostazione generale del lavoro

Il lavoro è organizzato in tre passaggi logici:

1. **Definizione del problema decisionale.** Il modello produce $\hat{p}(\text{BAD})$ e la policy trasforma la probabilità in un'approvazione/rifiuto tramite soglia ottimizzata su costi

e vincoli.

2. **Confronto modelli in chiave operativa.** Logistic Regression, Random Forest e XGBoost vengono valutati non solo con metriche di ranking (ROC-AUC, PR-AUC), ma soprattutto su KPI decisionali: costo, approval rate e *bad rate* tra gli approvati.
3. **Spiegabilità e robustezza.** La soluzione finale viene interpretata tramite driver (odds ratio) e verificata con una sensitivity analysis rispetto a ipotesi di costo e vincoli di volume.

L'obiettivo complessivo è produrre una policy **trasparente, difendibile e replicabile**, che espliciti le assunzioni economiche e renda visibile il trade-off tra crescita (volume) e prudenza (rischio).

2 Dataset

Il dataset utilizzato è *Statlog (German Credit Data)*, un benchmark storico per problemi di *credit scoring*. Il dataset raccoglie **1000** richieste di credito descritte da un set di variabili socio-economiche e contrattuali; l’obiettivo è classificare ciascun richiedente come *GOOD* (affidabile) o *BAD* (rischio di default/insolvenza). In questo report la classe positiva è definita come **BAD**, coerentemente con l’obiettivo di intercettare i casi rischiosi.

Il dataset è disponibile pubblicamente e viene spesso utilizzato per valutare pipeline di scoring in condizioni controllate.¹

Composizione e struttura. Le variabili includono sia attributi numerici (es. durata del prestito, importo, rate) sia categoriali (es. stato del conto corrente, storia creditizia, finalità del prestito, risparmi). Nel file utilizzato non risultano valori mancanti. La distribuzione del target è moderatamente sbilanciata (circa 70% *GOOD*, 30% *BAD*); per questo, oltre a ROC-AUC, nel confronto tra modelli vengono riportate anche metriche più informative sulla classe positiva (*BAD*), come la PR-AUC.

Nota di governance (variabili potenzialmente sensibili). Oltre alla versione “completa”, è stata considerata una variante **NO_SENS** che esclude due variabili potenzialmente sensibili presenti nel dataset (*personal_status_sex* e *foreign_worker*), con l’obiettivo di verificare se una policy più controllabile sul piano della governance sia ottenibile senza penalizzare le prestazioni operative.

Table 2: Snapshot del dataset utilizzato.

Osservazioni totali	1000
Feature totali	20
– numeriche	7
– categoriali	13
Target GOOD / BAD	700 / 300
Classe positiva	BAD
Valori mancanti	0

¹Dua, D. & Graff, C. (2019). UCI Machine Learning Repository: *Statlog (German Credit Data)*. University of California, Irvine.

3 Metodologia

3.1 Setup sperimentale e pipeline

L'analisi è organizzata come pipeline riproducibile, con separazione netta tra addestramento e valutazione per ridurre il rischio di *data leakage*. In particolare:

1. **Holdout split.** Suddivisione train/test (80/20) con stratificazione sul target, così da mantenere stabile la prevalenza di *BAD* tra i due insiemi.
2. **Pre-processing.** *One-hot encoding* delle variabili categoriali; gestione coerente delle feature tra train e test (stessa trasformazione appresa su train).
3. **Training modelli.** Addestramento e confronto di modelli con diversa complessità (Logistic Regression, Random Forest, XGBoost) sul solo training set.
4. **Ottimizzazione soglia.** Selezione del threshold t via cross-validation **solo su training**, minimizzando una funzione di costo asimmetrica sotto vincolo di *approval rate* minimo.
5. **Valutazione finale.** Applicazione della pipeline completa (modello + soglia) all'holdout test una sola volta per stimare performance fuori campione.

3.2 Regola decisionale e funzione di costo

Ogni modello produce una probabilità $\hat{p}(\text{BAD})$. La policy decisionale è:

$$\hat{p}(\text{BAD}) \geq t \Rightarrow \text{rifiuta (pred BAD)}, \quad \hat{p}(\text{BAD}) < t \Rightarrow \text{approva (pred GOOD)}.$$

La soglia t è scelta minimizzando un costo asimmetrico:

$$\text{Costo}(t) = c_{FP} \cdot FP(t) + c_{FN} \cdot FN(t), \quad \text{con } c_{FP} = 1, c_{FN} = 5,$$

dove *FP* sono *GOOD* rifiutati (costo opportunità) e *FN* sono *BAD* approvati (perdita attesa). Per evitare soluzioni troppo conservative, si impone anche un vincolo operativo:

$$\text{approval}(t) \geq 0.50.$$

Nota: il costo è definito in unità relative (scenario) e serve a confrontare policy sotto la stessa matrice dei costi, non a stimare perdite monetarie.

3.3 Confronto modelli: criterio operativo e governance

Il confronto tra modelli è impostato in modo coerente con l'uso reale: non si seleziona il modello “più accurato” in astratto, ma quello che genera la **migliore decisione** sotto vincoli.

Modelli valutati.

- **Logistic Regression:** baseline interpretabile; consente lettura diretta dei driver tramite coefficienti e *odds ratio*.
- **Random Forest:** modello non lineare robusto, in grado di catturare interazioni senza specificarle esplicitamente.
- **XGBoost:** boosting competitivo su dati tabellari, spesso efficace nel catturare non linearità e pattern complessi.

Perché non basta ROC-AUC. ROC-AUC e PR-AUC misurano principalmente la capacità di ordinare i clienti per rischio, ma una banca deve prendere una decisione binaria con vincoli di volume. Per questo, la selezione finale si basa su KPI decisionali:

- **costo totale** della policy ($FP + 5 \cdot FN$);
- **approval rate** (volume approvato) e **bad rate tra approvati** (qualità del portafoglio approvato);
- **stabilità** della soglia selezionata in cross-validation.

Variante NO_SENS. In parallelo alla versione “completa”, viene valutata una versione che esclude due variabili potenzialmente sensibili (*personal_status_sex*, *foreign_worker*). L’obiettivo è verificare se una policy più controllabile sul piano della governance sia ottenibile senza peggiorare la performance operativa.

4 Risultati

4.1 Confronto tra modelli (holdout test)

In questa sezione confronto i modelli non “in astratto”, ma nel modo in cui verrebbero usati davvero: una probabilità di default viene trasformata in una decisione (approva / rifiuta) tramite una soglia t . Per rendere il confronto corretto, ogni modello viene valutato con la stessa logica decisionale:

- la soglia t è selezionata via cross-validation sul **solo training set**;
- l’obiettivo è minimizzare il costo $FP + 5 \cdot FN$ (costi asimmetrici);
- viene imposto un vincolo operativo: *approval rate* $\geq 50\%$;
- il test set (holdout) è usato **solo** per la valutazione finale.

La Tabella 3 riporta sia metriche operative (costo, approval, qualità del portafoglio approvato) sia metriche di ranking (ROC-AUC, PR-AUC). Le metriche di ranking sono utili per capire se il modello “ordina bene” i clienti, ma non dicono da sole quale policy conviene adottare: nel credito, la differenza la fa l’impatto operativo della soglia scelta.

Table 3: Confronto modelli su holdout test (soglia da CV, vincolo approval $\geq 50\%$).

Modello	Costo	Approval	BAD appr.	Recall BAD	ROC-AUC	PR-AUC
LogReg_NO_SENS	96	0.52	0.096	0.833	0.803	0.651
LogReg_ALL	105	0.505	0.109	0.817	0.804	0.635
Random Forest	112	0.50	0.120	0.800	0.808	0.671
XGBoost	114	0.49	0.122	0.800	0.796	0.658

Interpretazione. Il risultato non è “il modello con AUC più alta”, ma quello che produce la miglior decisione secondo il criterio stabilito. In questo scenario, **LogReg_NO_SENS** ottiene il **costo minimo** sul test mantenendo il vincolo di volume (approval $\approx 52\%$) e una quota di clienti *BAD* tra gli approvati più bassa rispetto alle alternative. Inoltre, la rimozione delle variabili sensibili migliora la coerenza con principi di governance senza penalizzare le prestazioni operative (anzi, qui migliora il costo).

4.2 Performance operativa del modello finale

Una volta scelto il modello, la domanda giusta non è “quanto è alta l’AUC”, ma *che cosa succede davvero quando lo uso per decidere chi approvare*. Per questo motivo la valutazione è centrata su indicatori operativi: costo totale, volumi approvati e qualità del portafoglio che finisce “in pancia” all’azienda.

Il punto chiave è che la probabilità stimata $\hat{p}(\text{BAD})$ diventa una decisione tramite una soglia t . Quella soglia determina contemporaneamente:

- quante richieste vengono approvate (*approval rate*);
- quante posizioni rischiose vengono fatte passare (*BAD* tra gli approvati);
- quanti *BAD* vengono bloccati (*recall BAD*).

Questi tre aspetti vanno letti insieme: è inutile “bloccare tutti i *BAD*” se poi approvi troppo poco e non fai business; ed è inutile “approvare tanto” se poi il portafoglio approvato è pieno di casi rischiosi.

La Tabella 4 riporta i numeri finali su holdout test per il modello scelto (**LogReg_NO_SENS**, $t = 0.25$). In sintesi: si approva circa metà delle richieste (0.52), e tra gli approvati la quota di *BAD* è circa 9.6%. Questo è il dato che, in un contesto reale, si traduce in “qualità del portafoglio approvato”. Il *recall BAD* (0.833) indica invece che la policy intercetta la maggior parte dei casi rischiosi, riducendo l’esposizione a default non bloccati.

Table 4: KPI operativi su holdout test (**LogReg_NO_SENS**, $t = 0.25$).

Costo totale ($FP + 5 \cdot FN$)	96
Costo per osservazione	0.48
Approval rate (pred <i>GOOD</i>)	0.52
Bad rate tra approvati	0.096
Recall <i>BAD</i>	0.833
Precision <i>BAD</i>	0.521
TN / FP / FN / TP	94 / 46 / 10 / 50

La Confusion Matrix (Figura 1) aiuta a capire *come* si compone quel costo. I falsi negativi ($FN = 10$, cioè *BAD* approvati) sono gli errori più pericolosi: sono esattamente quelli che generano perdita attesa nel mondo reale. I falsi positivi ($FP = 46$, *GOOD* rifiutati) rappresentano invece un costo opportunità: business perso, ma rischio evitato. Questa distinzione è il motivo per cui si usa una cost matrix e non una metrica unica “generica”.

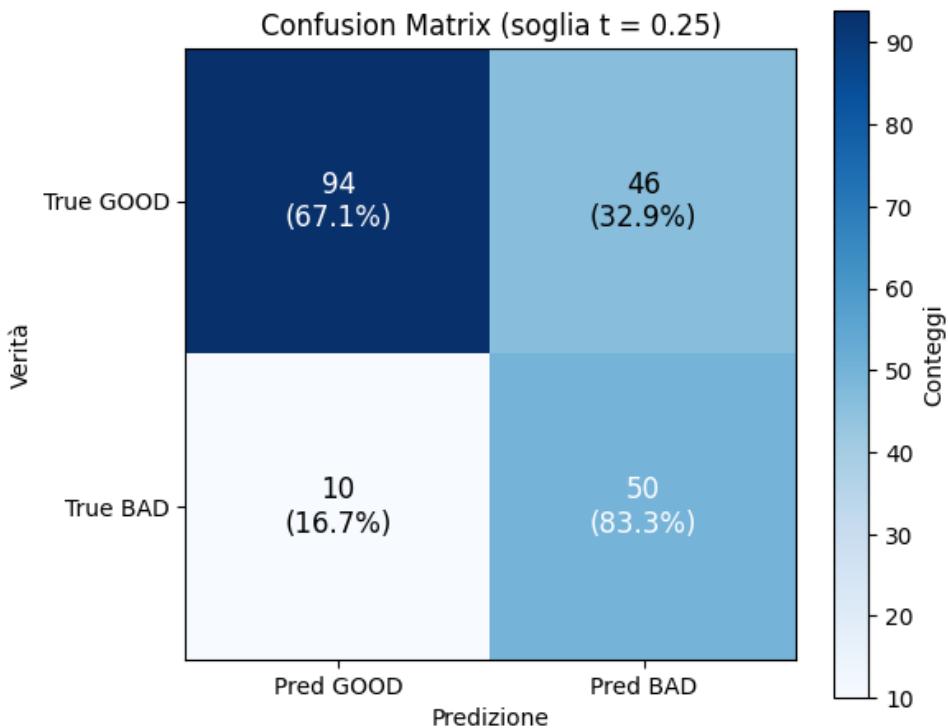


Figure 1: Confusion matrix su holdout test (LogReg_NO_SENS, $t = 0.25$).

4.3 Scelta della soglia: evidenza del trade-off costo–volume

La soglia t è la parte più “decisionale” di tutto il progetto. Due modelli con performance simili possono portare a policy completamente diverse se scegli t in modo sbagliato. Per questo qui non mi limito a dire “ho scelto $t = 0.25$ ”, ma mostro perché quella soglia è sensata.

L’idea è semplice: variando t , cambiano FP e FN . E, di conseguenza, cambia il costo:

$$\text{Costo}(t) = 1 \cdot FP(t) + 5 \cdot FN(t).$$

Con soglie più basse, la policy è più severa: rifiuta più spesso, quindi tende a ridurre i *BAD* approvati (meno FN), ma abbassa i volumi approvati. Con soglie più alte, la policy è più permissiva: aumenta l’approval rate, ma cresce il rischio che passino più *BAD* (più FN).

Il vincolo $\text{approval} \geq 50\%$ è stato inserito apposta per evitare un caso tipico: ottimizzare il costo “puro” può portare a soluzioni troppo conservative, che minimizzano le perdite solo perché approvano pochissimo. In un contesto reale, quella non è una soluzione: è un rifiuto del problema.

La Figura 2 rende visibile questo equilibrio. La soglia finale ($t = 0.25$) è il punto che minimizza il costo *dentro* l’area accettabile dal punto di vista operativo (volumi). Questo trasforma la scelta del threshold da “impostazione a caso” a policy motivata e difendibile.

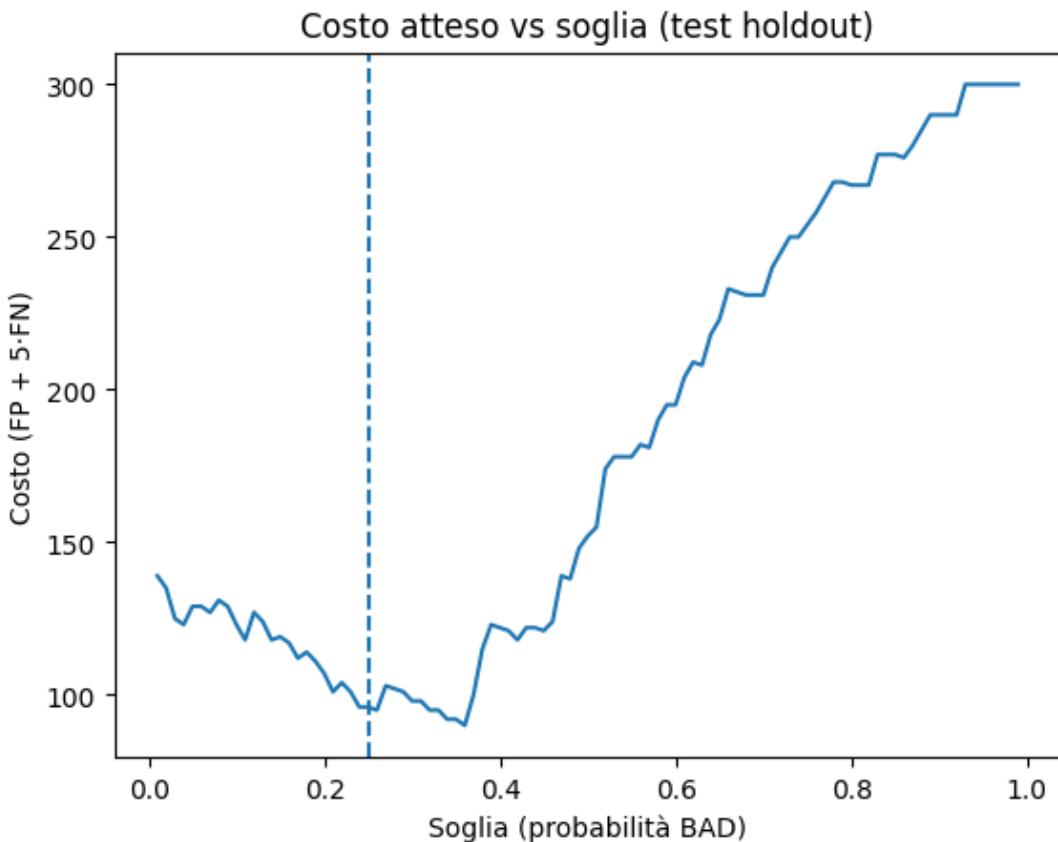


Figure 2: Costo vs soglia: selezione del threshold con vincolo di approvazione.

4.4 Interpretabilità: cosa guida il rischio (driver)

Un modello di *credit scoring* deve essere non solo accurato, ma anche **comprendibile**. Se non è chiaro perché una richiesta viene considerata rischiosa, diventa difficile fidarsi della policy, spiegarla a stakeholder non tecnici e, soprattutto, migliorarla nel tempo. Per questo motivo l'interpretabilità qui non è un esercizio “accademico”: è parte integrante della qualità della soluzione.

Dato che il modello finale è una Logistic Regression, l'interpretazione è relativamente diretta:

- un coefficiente $\beta > 0$ aumenta la probabilità stimata di *BAD* (a parità delle altre variabili);
- un coefficiente $\beta < 0$ la riduce.

Per rendere l'effetto più intuitivo, i coefficienti vengono convertiti in *odds ratio* $\exp(\beta)$: valori > 1 indicano un aumento del rischio, valori < 1 un effetto protettivo.

La Figura 3 mostra i driver principali del rischio con etichette “in chiaro” (mappatura dei codici Axx). Il senso pratico di questo grafico è duplice:

- **Spiegare la decisione.** Permette di collegare le scelte di approvazione/rifiuto a condizioni specifiche e comunicabili (es. caratteristiche del conto, risparmi, storia creditizia, finalità del prestito).
- **Verificare la solidità del modello.** Se un driver appare “fortissimo” ma è legato a una categoria rarissima, è più probabile che sia rumore statistico che informazione robusta.

Cautela sulle categorie rare. Quando una categoria ha pochissimi casi, anche un singolo default in più o in meno può alterare molto il coefficiente stimato. Per questo motivo i driver associati a livelli rari vengono interpretati come segnali esplorativi, mentre le conclusioni principali si basano su effetti che riguardano categorie frequenti e stabili nel campione.

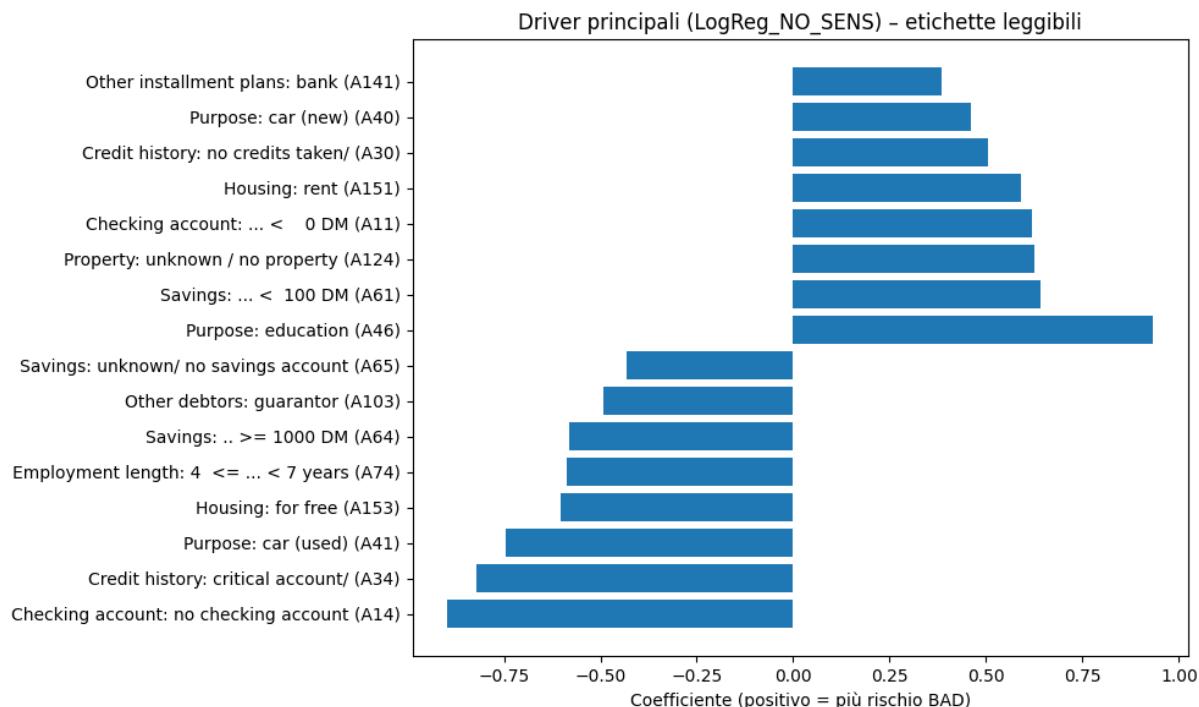


Figure 3: Driver principali del rischio (LogReg_NO_SENS): effetti verso *BAD* o *GOOD* (odds ratio).

4.5 Robustezza: sensitivity su costi e vincoli

Una policy *cost-sensitive* è utile solo se resta sensata quando cambiano le ipotesi economiche. Nel mondo reale, infatti, i “costi” degli errori non sono numeri fissi: dipendono dal contesto macroeconomico, dal tipo di prodotto (prestito personale, revolving, mutuo), dai processi di recupero crediti, dal pricing e persino dagli obiettivi commerciali del periodo. Per questo motivo sarebbe poco serio presentare una soglia ottimale come se fosse “vera” in assoluto. Quello che si può fare in modo corretto è valutare *quanto* la policy cambia al variare di ipotesi ragionevoli: esattamente lo scopo della sensitivity analysis.

La sensitivity è costruita attorno a due scelte che, in pratica, rappresentano decisioni di business:

- **Asimmetria dei costi ($c_{FN} : c_{FP}$)**. Qui si controlla quanto voglio essere severo nel ridurre i *BAD* approvati (falsi negativi), cioè i casi che generano perdite attese. Un aumento di c_{FN} rende la policy più prudente: tenderà a rifiutare più spesso pur di ridurre l'esposizione al default.
- **Vincolo di volume (approval minimo)**. Questo riflette un vincolo operativo tipico: non basta minimizzare il rischio, bisogna anche mantenere un livello minimo di erogazioni. A parità di modello, imporre un approval più alto forza la soglia verso decisioni più permissive.

La Figura 4 mostra la soglia selezionata via cross-validation al variare di (i) $c_{FN} : c_{FP}$ e (ii) del vincolo di approval. Il grafico è pensato per rispondere a una domanda semplice: “Se cambiano le regole del gioco, la policy cambia di poco o cambia drasticamente?”

Risultato principale: stabilità nel regime operativo rilevante. Nel regime operativo di maggiore interesse per questo progetto ($approval \geq 50\%$), la soglia ottimale tende a stabilizzarsi intorno a $t \approx 0.25$ quando $c_{FN} : c_{FP} \geq 3 : 1$. Questa è un'evidenza importante perché significa che la policy non è appesa a un singolo valore arbitrario (ad esempio 5 : 1): rimane coerente anche se l'azienda rivaluta la severità verso i falsi negativi entro un intervallo plausibile. In altre parole, il modello produce una regola decisionale *robusta* rispetto a ipotesi economiche ragionevoli.

Il grafico rende anche esplicito il prezzo da pagare per policy più aggressive:

- se si impone un approval molto alto, la soglia deve aumentare (policy più permissiva) e cresce il rischio che aumentino i *BAD* approvati;
- se si riduce troppo l'asimmetria ($c_{FN} : c_{FP}$ vicino a 1 : 1), la policy diventa meno sensibile all'errore più critico (default approvati) e tende a privilegiare volume/precisione complessiva rispetto alla protezione dal rischio.

Questo non è “giusto” o “sbagliato”: è una scelta strategica. L’obiettivo della sensitivity è rendere visibile il trade-off e permettere una decisione consapevole.

Questa analisi non stima i costi reali e non pretende di identificare effetti causali: è un esercizio di *policy design* basato su scenari. Il suo valore sta nel trasformare assunzioni implicite (quanto pesa un default? quante richieste devo approvare?) in parametri esplicativi e verificare la stabilità della regola decisionale.

Implicazione pratica. La sensitivity analysis fornisce una “mappa di controllo” della policy: se cambiano le priorità aziendali (più crescita o più prudenza) oppure cambia il contesto (aumento delle perdite attese), è chiaro come adeguare c_{FN} , il vincolo di approval e, di conseguenza, la soglia t . Questo rende la soluzione non solo accurata, ma governabile nel tempo.

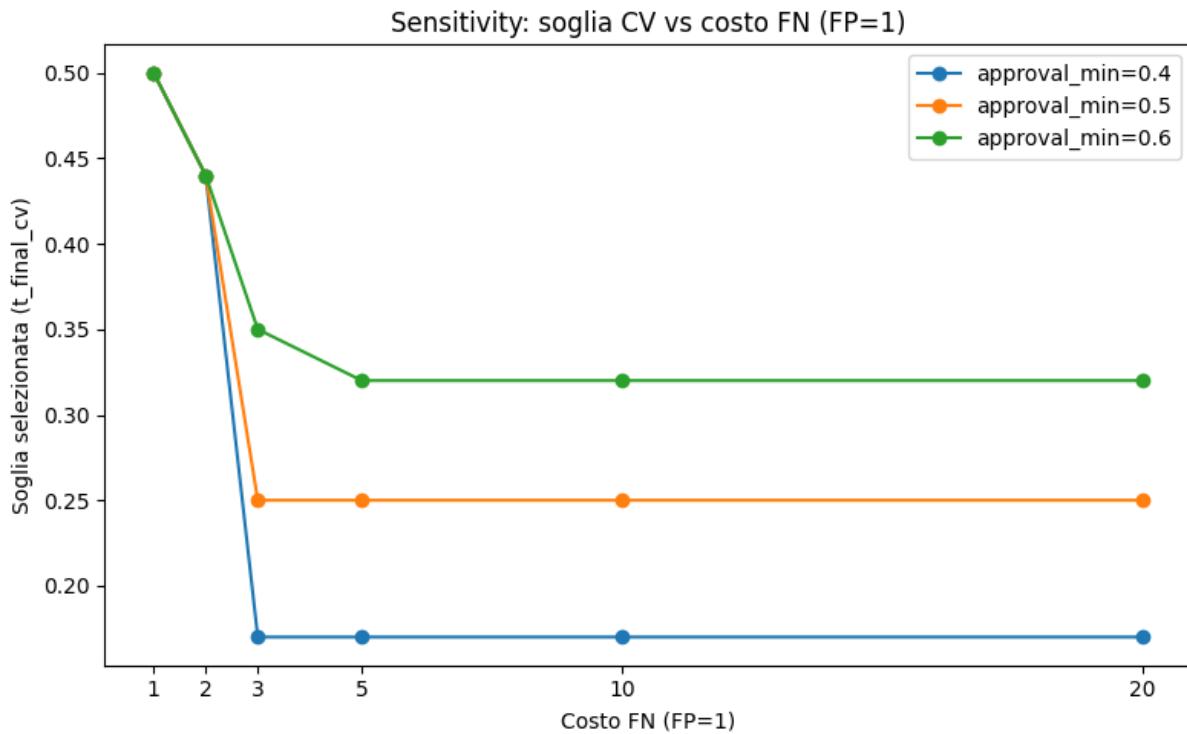


Figure 4: Sensitivity: soglia selezionata vs costo dei falsi negativi (FP=1) per diversi vincoli di approval.

5 Conclusioni

5.1 Sintesi dei risultati

Il progetto ha costruito una **policy decisionale** di approvazione del credito, non solo un modello predittivo. La probabilità di default stimata viene trasformata in una decisione operativa (approva/rifiuta) tramite una soglia selezionata in modo esplicito e replicabile, coerente con un criterio economico (costi asimmetrici) e con un vincolo minimo sui volumi approvati.

Nel confronto tra modelli, la soluzione che offre il miglior equilibrio tra impatto operativo e spiegabilità è **LogReg_NO_SENS** con soglia $t = 0.25$. In particolare, la policy finale:

- mantiene un **approval rate** vicino al 50%, assicurando volumi di erogazione non marginali;
- contiene la quota di **BAD tra gli approvati**, migliorando la qualità del portafoglio approvato;
- intercetta la maggior parte dei casi rischiosi, come evidenziato dal **recall BAD** elevato;
- mostra **stabilità** rispetto a variazioni ragionevoli delle ipotesi di costo, come indicato dalla sensitivity analysis.

5.2 Raccomandazioni operative

Sulla base delle evidenze prodotte, le raccomandazioni più concrete sono:

- **Usare una soglia ottimizzata, non arbitraria.** La selezione del threshold tramite minimizzazione del costo atteso (con vincolo di approval) fornisce una regola decisionale difendibile e direttamente collegata agli obiettivi operativi.
- **Valutare la performance in termini decisionali.** Oltre a metriche di ranking, è essenziale monitorare indicatori operativi come costo totale, approval rate e *bad rate* tra gli approvati, perché descrivono immediatamente il comportamento della policy.
- **Mantenere interpretabilità e controllo.** La lettura dei driver tramite *odds ratio* consente di spiegare la decisione e di verificare che i segnali dominanti siano plausibili e non guidati da categorie rare o instabili.
- **Esplicitare le assunzioni economiche.** La cost matrix e i vincoli di volume rappresentano scelte di business: renderle esplicite (e verificarne la stabilità) migliora la trasparenza e la governabilità della policy.

6 Limiti e assunzioni

- **Dataset benchmark e dimensione campionaria.** Il German Credit è un dataset standard di riferimento (1000 osservazioni): utile per validare metodologia e pipeline, ma con limiti intrinseci di rappresentatività rispetto a portafogli reali e a dinamiche macroeconomiche.
- **Costi degli errori come scenario.** La funzione di costo $FP + 5 \cdot FN$ non rappresenta una stima “vera” delle perdite economiche, ma formalizza un’asimmetria plausibile tra errori. Il valore aggiunto del progetto è rendere questa scelta esplicita e verificare la stabilità della policy tramite sensitivity analysis.
- **Categorie rare.** Alcuni livelli categoriali hanno frequenza molto bassa. In questi casi i coefficienti possono essere instabili; per questo l’interpretazione dei driver privilegia effetti frequenti e coerenti, evitando conclusioni forti su categorie rare.
- **Predizione vs causalità.** Le relazioni individuate dai driver sono associazioni utili per la decisione predittiva, ma non devono essere lette come effetti causali.

A Note tecniche: metriche e frammenti di codice essenziali

A.1 Metriche (definizioni sintetiche)

- **ROC-AUC.** Capacità di ordinare *BAD* sopra *GOOD* al variare della soglia (metrica di ranking).
 - **PR-AUC.** Area sotto la curva Precision–Recall; informativa quando la classe positiva (*BAD*) è meno frequente.
 - **Recall BAD.** $TP/(TP + FN)$: quota di *BAD* intercettati; riduce i *BAD* approvati.
 - **Precision BAD.** $TP/(TP+FP)$: quota di predetti *BAD* che sono effettivamente *BAD*.
 - **Brier score.** $\frac{1}{n} \sum_i (\hat{p}_i - y_i)^2$: qualità delle probabilità (calibrazione).
 - **Confusion matrix.** TN, FP, FN, TP : scomponi successi ed errori della policy alla soglia scelta.

A.2 Codice essenziale (riproducibilità delle scelte chiave)

I frammenti seguenti documentano le scelte più importanti: (i) funzione di costo, (ii) selezione della soglia con vincolo di *approval*, (iii) schema di cross-validation per stimare t sul training, (iv) estrazione dei driver tramite coefficienti e *odds ratio*. L'intero workflow è riproducibile tramite gli script allegati.

A.2.1 Funzione di costo e ricerca della soglia con vincolo di approval

```

15     continue
16     cost, fp, fn = expected_cost(y_true, y_pred_bad, c_fp,
17                                    c_fn)
18     if (best is None) or (cost < best["cost"]):
19         best = {"t": t, "cost": cost, "approval": approval,
20                  "fp": fp, "fn": fn}
21 return best

```

A.2.2 Cross-validation: soglia stimata sul training (anti-leakage)

```

1 import numpy as np
2 from sklearn.model_selection import StratifiedKFold
3
4 skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
5 fold_results = []
6
7 for tr_idx, va_idx in skf.split(X_train, y_train):
8     X_tr, X_va = X_train.iloc[tr_idx], X_train.iloc[va_idx]
9     y_tr, y_va = y_train.iloc[tr_idx], y_train.iloc[va_idx]
10
11    model.fit(X_tr, y_tr)
12    p_bad_va = model.predict_proba(X_va)[:, 1]
13
14    best = find_best_threshold(
15        p_bad_va, y_va,
16        c_fp=1, c_fn=5,
17        approval_min=0.50
18    )
19    fold_results.append(best)
20
21 t_final = float(np.median([r["t"] for r in fold_results]))

```

A.2.3 Driver del rischio: coefficienti e odds ratio (Logistic Regression)

```

1 import numpy as np
2 import pandas as pd
3
4 coef = model.named_steps["clf"].coef_.ravel()
5 feat = model.named_steps["preprocess"].get_feature_names_out()
6
7 drivers = pd.DataFrame({"feature": feat, "coef": coef})
8 drivers["odds_ratio"] = np.exp(drivers["coef"])
9
10 top_risk = drivers.sort_values("coef", ascending=False).head(10)
11 top_prot = drivers.sort_values("coef", ascending=True).head(10)

```