




Analisi e benchmark di classificazione tramite three-way-decision

AA 2024/2025

 Ruben Tenderini

r.tenderini@campus.unimib.it

Università degli Studi di Milano-Bicocca

Indice

1	Introduzione	1
2	Fondamenti Teorici	2
2.1	Origini e motivazioni	2
2.2	Struttura concettuale: Trisecting e Acting	2
2.2.1	Trisecting	2
2.2.2	Acting	3
2.3	Approcci formali e varianti	3
2.3.1	Approccio decision-theoretic	3
2.3.2	Approcci basati su Rough Sets	3
2.3.3	Varianti e generalizzazioni	3
2.4	Applicazioni nella classificazione con astensione	4
2.5	Vantaggi e sfide	4
2.6	Sintesi	4
3	Analisi dei Dataset	5
3.1	UCI Bank Marketing Dataset	5
3.1.1	Descrizione generale	5
3.1.2	Feature	6
3.1.3	Distribuzione della variabile target	6
3.2	UCI Adult Census Income Dataset	6
3.2.1	Descrizione generale	6
3.2.2	Feature	7
3.2.3	Distribuzione della variabile target	7
4	Metodologia Sperimentale	8
4.1	Configurazione dell'ambiente di test	8
4.2	Classificatori	8
4.3	Strumenti di misurazione	9
4.3.1	Report CSV	9
4.3.2	Visualizzazioni e grafici	9
4.4	Architettura del Software	10
4.4.1	Pipeline di esecuzione	10
4.4.2	Struttura del codice	10
5	Risultati Sperimentali	11
5.1	Bank Marketing	11
5.1.1	Metriche principali	11
5.1.2	Istogramma delle probabilità	12
5.1.3	Matrici di confusione	13
5.1.4	Curve ROC	13
5.1.5	Diagrammi di Calibrazione	14
5.1.6	Grafici Coverage-Accuracy	15

5.1.7	Grafici Precision-Recall	16
5.2	Adult Census Income	16
5.2.1	Metriche principali	16
5.2.2	Istogramma delle probabilità	17
5.2.3	Matrici di confusione	18
5.2.4	Curve ROC	19
5.2.5	Diagrammi di Calibrazione	20
5.2.6	Grafici Coverage-Accuracy	21
5.2.7	Grafici Precision-Recall	21
6	Conclusione e Sviluppi Futuri	23
6.1	Sintesi dei Risultati	23
6.2	Sviluppi futuri	23
A	Codice Python	25
A.1	main.py	25
A.2	functions.py	30
	Bibliografia	39

Capitolo 1

Introduzione

Questo progetto affronta il tema delle Three-Way Decision (3WD) applicate alla classificazione, con l'obiettivo di integrare il concetto di astensione nei modelli decisionali automatici. L'idea centrale è permettere a un classificatore di non prendere una decisione quando il livello di incertezza è troppo elevato, richiedendo invece una revisione o un intervento umano.

L'obiettivo principale è progettare, implementare e confrontare diversi metodi di classificazione che incorporino l'astensione, valutandone l'impatto in termini di accuratezza, tasso di astensione e costi decisionali complessivi. Questo approccio risulta particolarmente utile in contesti dove un errore può avere conseguenze critiche, come nella diagnosi medica, nel rilevamento di frodi o nei sistemi di supporto alle decisioni.

La relazione è strutturata come segue:

- **Capitolo 2 – Fondamenti teorici:** introduzione ai principi della three-way decision e alle basi matematiche del modello.
- **Capitolo 3 – Analisi dei dataset:** descrizione dei dataset utilizzati e delle loro caratteristiche principali.
- **Capitolo 4 – Metodologia sperimentale:** presentazione dell'ambiente di lavoro, degli strumenti e delle tecniche impiegate.
- **Capitolo 5 – Risultati sperimentali:** analisi dei risultati ottenuti.
- **Capitolo 6 – Conclusioni e sviluppi futuri:** sintesi delle osservazioni principali e possibili direzioni per lavori successivi.

Per trasparenza e riproducibilità, il codice sorgente completo del progetto è disponibile su GitHub all'indirizzo:

<https://github.com/Ruben-2828/three-way-decision-ml>

Capitolo 2

Fondamenti Teorici

In questo capitolo vengono illustrati i principi fondamentali del paradigma delle *Three-Way Decisions* (3WD), la loro origine, i concetti chiave e le principali applicazioni nel contesto della classificazione con astensione.

2.1 Origini e motivazioni

Il concetto di *three-way decision* nasce come estensione dei modelli decisionali binari tradizionali, introducendo una terza opzione di non impegno (*defer* o *abstain*) per gestire situazioni di incertezza. Piuttosto che forzare una decisione tra “accetta” e “rifiuta”, il modello 3WD consente di sospendere il giudizio quando il livello di fiducia non è sufficiente per una scelta sicura.

Il paradigma è stato formalizzato da Yiyu Yao all'interno della teoria degli insiemi approssimati (*rough set theory*), collegandolo a un processo decisionale basato sull'analisi del rischio [9]. Yao definisce il modello in due fasi principali: *trisecting* (suddivisione in tre regioni decisionali) e *acting* (scelta dell'azione corrispondente) [8].

Questo approccio si è rapidamente diffuso in vari campi, come l'analisi dei conflitti, il supporto alle decisioni e la classificazione automatica, dove l'incertezza rappresenta un elemento critico [10].

2.2 Struttura concettuale: Trisecting e Acting

Il modello 3WD si articola in due fasi complementari:

2.2.1 Trisecting

La prima fase consiste nel dividere l'universo degli oggetti in tre regioni disgiunte, definite rispetto a due soglie α e β , con $\alpha < \beta$:

- **Regione positiva** (*accept region*): gli oggetti con alta probabilità di appartenere alla classe positiva ($P(Y = 1|x) \geq \beta$);
- **Regione negativa** (*reject region*): gli oggetti con bassa probabilità ($P(Y = 1|x) \leq \alpha$);
- **Regione di confine o indeterminata** (*defer region*): gli oggetti per cui $\alpha < P(Y = 1|x) < \beta$.

La scelta delle soglie può derivare da criteri probabilistici, regole di minimizzazione del rischio o approcci basati su insiemi approssimati probabilistici [3].

2.2.2 Acting

Una volta definita la trisezione, il passo successivo consiste nel determinare l'azione da intraprendere per ogni regione:

- accettare la decisione positiva;
- rifiutare la decisione (classe negativa);
- astenersi o rimandare la decisione a un livello superiore (ad esempio, un esperto umano o un modello più accurato).

Questo modello, detto *trisecting-and-acting*, fornisce una visione sistematica e modulare della presa di decisione incerta [8].

2.3 Approcci formali e varianti

2.3.1 Approccio decision-theoretic

Un approccio formale alla 3WD si basa sulla minimizzazione del rischio atteso, utilizzando funzioni di perdita (*loss functions*) per ciascuna azione possibile. Sia $P(Y = 1|x)$ la probabilità a posteriori e siano L_a , L_r , e L_d le perdite associate rispettivamente ad accettare, rifiutare e astenersi. La decisione ottimale per ogni istanza x è quella che minimizza il rischio atteso:

$$R(a_i|x) = \sum_y L(a_i, y)P(y|x)$$

Si sceglie quindi l'azione a_i tale che $R(a_i|x)$ sia minima [8].

Questo schema consente di derivare automaticamente le soglie α e β a partire dalle funzioni di perdita, rendendo la metodologia coerente con la teoria classica della decisione.

2.3.2 Approcci basati su Rough Sets

Il legame tra 3WD e teoria degli insiemi approssimati è profondo: in un modello *rough set*, ogni concetto viene rappresentato tramite un'*approssimazione inferiore*, un'*approssimazione superiore* e una *zona di confine*. La *three-way decision* fornisce una generalizzazione di questo schema, sostituendo la visione puramente insiemistica con una interpretazione decisionale [9, 8].

2.3.3 Varianti e generalizzazioni

Diversi autori hanno proposto estensioni della 3WD in contesti fuzzy, linguistici o multi-criterio. Ad esempio, modelli fuzzy a tre vie consentono di rappresentare gradi di appartenenza incerti [4], mentre versioni multi-criterio applicano la 3WD a problemi di decisione complessi [7]. In ambito di apprendimento automatico, Campagner et al. [1] hanno introdotto il concetto di *three-way classification*, in cui un classificatore probabilistico può astenersi in presenza di ambiguità.

2.4 Applicazioni nella classificazione con astensione

L'applicazione della 3WD alla classificazione con astensione rappresenta un'estensione naturale del paradigma. In un modello di classificazione probabilistica, la decisione viene presa solo quando la confidenza supera una certa soglia, mentre nei casi ambigui il modello si astiene.

Questo approccio consente di ridurre il numero di errori nei casi difficili, trasferendo la responsabilità delle istanze incerte a un processo di revisione umana o a un modello più complesso [1, 6].

Le applicazioni più comuni si trovano in:

- diagnostica medica, dove un errore può avere conseguenze gravi;
- rilevamento di frodi e sicurezza informatica;
- sistemi di supporto alle decisioni in ambienti ad alta incertezza.

2.5 Vantaggi e sfide

Vantaggi principali:

- maggiore controllo sul rischio decisionale;
- capacità di rappresentare e gestire esplicitamente l'incertezza;
- possibilità di integrazione con modelli di costo realistici.

Sfide principali:

- scelta ottimale delle soglie α e β ;
- bilanciamento tra tasso di astensione e accuratezza complessiva;
- gestione dei costi associati alle istanze astentive.

2.6 Sintesi

La teoria delle *Three-Way Decisions* offre un quadro concettuale robusto per integrare l'astensione nei processi decisionali e di classificazione. Nei capitoli successivi verranno descritti i dataset utilizzati e le metodologie sperimentali impiegate per valutare l'efficacia di diversi approcci basati su 3WD.

Capitolo 3

Analisi dei Dataset

In questo capitolo vengono descritti i dataset utilizzati per lo studio sperimentale.

3.1 UCI Bank Marketing Dataset

3.1.1 Descrizione generale

Il dataset *Bank Marketing* proviene da una campagna di marketing diretta di una banca portoghese, volta a promuovere la sottoscrizione di un deposito a termine [5]. Ogni record rappresenta un cliente contattato durante la campagna e include informazioni socio-demografiche, dettagli sul contatto e caratteristiche della campagna stessa.

- **Fonte:** UCI Machine Learning Repository
- **Numero di istanze:** 45,211
- **Numero di attributi:** 17 (inclusa la variabile target)
- **Tipo di problema:** Classificazione binaria
- **Variabile target:** y – yes se il cliente ha sottoscritto il deposito, no altrimenti

3.1.2 Feature

Tabella 3.1: Attributi del dataset *Bank Marketing*

Nome attributo	Tipo	Descrizione
age	Numerico	Età del cliente
job	Categoriale	Tipo di lavoro
marital	Categoriale	Stato civile
education	Categoriale	Livello di istruzione
default	Categoriale	Presenza di default sul credito
balance	Numerico	Bilancio medio annuale sul conto
housing	Categoriale	Presenza di mutuo sulla casa
loan	Categoriale	Presenza di prestito personale
contact	Categoriale	Tipo di contatto
day_of_week	Numerico	Giorno dell'ultimo contatto
month	Categoriale	Mese dell'ultimo contatto
duration	Numerico	Durata del contatto in secondi
campaign	Numerico	Numero di contatti effettuati durante la campagna
pdays	Numerico	Giorni trascorsi dall'ultimo contatto
previous	Numerico	Numero di contatti precedenti
poutcome	Categoriale	Esito della campagna precedente
y	Categoriale	Variabile target: yes, no

3.1.3 Distribuzione della variabile target

La variabile target è fortemente sbilanciata:

- **no:** circa l'88% delle istanze
- **yes:** circa il 12% delle istanze

3.2 UCI Adult Census Income Dataset

3.2.1 Descrizione generale

Il dataset *Adult Census Income*, noto anche come *Adult* o *Census Income*, proviene dal censimento statunitense del 1994 e viene spesso impiegato per predire se un individuo guadagna più o meno di \$50.000 annui [2].

- **Fonte:** UCI Machine Learning Repository
- **Numero di istanze:** 48,842
- **Numero di attributi:** 14 (inclusa la variabile target)
- **Tipo di problema:** Classificazione binaria
- **Variabile target:** income – $\leq 50K$ oppure $> 50K$

3.2.2 Feature

Tabella 3.2: Attributi del dataset *Adult Census Income*

Nome attributo	Tipo	Descrizione
age	Numerico	Età dell'individuo
workclass	Categoriale	Tipo di impiego
fnlwgt	Numerico	Peso di rappresentatività dell'individuo
education	Categoriale	Livello di istruzione
education-num	Numerico	Anni di istruzione
marital-status	Categoriale	Stato civile
occupation	Categoriale	Tipo di occupazione
relationship	Categoriale	Relazione familiare
race	Categoriale	Etnia dichiarata
sex	Categoriale	Sesso biologico
capital-gain	Numerico	Guadagno da capitale
capital-loss	Numerico	Perdita da capitale
hours-per-week	Numerico	Ore lavorate a settimana
native-country	Categoriale	Paese di origine
income	Categoriale	Variabile target: $\leq 50K$, $> 50K$

L'attributo fnlwgt (final weight) non viene utilizzato come feature predittiva, in quanto non apporta informazioni rilevanti sulla variabile target, ma indica solamente il numero di persone rappresentate da quell'individuo del campione.

3.2.3 Distribuzione della variabile target

Il dataset presenta uno sbilanciamento tra le due classi:

- $\leq 50K$: circa il 76% delle istanze
- $> 50K$: circa il 24% delle istanze

Durante la fase di preprocessing, è stato poi eseguito un bilanciamento delle classi, tramite undersampling, per poter addestrare i modelli anche con un dataset bilanciato.

Capitolo 4

Metodologia Sperimentale

In questo capitolo vengono descritte le modalità di sperimentazione adottate nel progetto, in particolare la configurazione dell'ambiente di test, i classificatori utilizzati e gli strumenti di misurazione impiegati per valutare le prestazioni dei modelli secondo l'approccio 3WD.

4.1 Configurazione dell'ambiente di test

Gli esperimenti sono stati condotti utilizzando il linguaggio **Python 3.12** e le seguenti librerie principali:

- **scikit-learn**: per l'implementazione dei modelli di classificazione, la divisione del dataset in insiemi di addestramento e test, e il calcolo delle metriche di performance;
- **numpy**: per la gestione efficiente di array e operazioni numeriche;
- **matplotlib**: per la generazione dei grafici di analisi e confronto dei risultati;
- **pandas**: per la manipolazione dei dataset e la gestione di file CSV;
- **os**: per la gestione dei percorsi e delle directory di output;

Il sistema su cui sono stati eseguiti i test presenta le seguenti caratteristiche hardware e software:

- **Hardware**: *CPU AMD Ryzen 5 5600H, 16GB RAM*
- **Sistema operativo**: *Windows 11*

4.2 Classificatori

Per la fase di classificazione sono stati implementati due modelli principali:

- **Random Forest (RF)**: modello basato su un insieme di alberi decisionali addestrati su sottoinsiemi casuali dei dati e delle feature, robusto al rumore e adatto a dataset con variabili miste (numeriche e categoriali). Nel progetto è stato utilizzato un modello con 200 alberi (`n_estimators=200`).
- **Multi-Layer Perceptron (MLP)**: rete neurale feed-forward con due strati nascosti rispettivamente da 128 e 64 neuroni, funzione di attivazione ReLU e ottimizzatore Adam. È stato abilitato l'*early stopping* per evitare fenomeni di overfitting e migliorare la stabilità del training.

Entrambi i modelli producono, per ogni istanza, una probabilità di appartenenza alla classe positiva. Tali probabilità vengono poi elaborate attraverso due soglie di decisione:

- β : soglia superiore di accettazione (classificazione positiva);
- α : soglia inferiore di rifiuto (classificazione negativa).

Le istanze per cui la probabilità predetta ricade nell'intervallo intermedio ($\alpha < p < \beta$) vengono considerate *indecise* e assegnate alla regione di astensione (*defer region*).

4.3 Strumenti di misurazione

Per l'analisi dei risultati è stato implementato un salvataggio delle metriche principali in un file CSV insieme a visualizzazioni grafiche di supporto.

4.3.1 Report CSV

Il report dei risultati sperimentali viene salvato in formato CSV e contiene le seguenti colonne:

- **model** — nome del classificatore utilizzato;
- **accuracy** — accuratezza complessiva del modello;
- **precision, recall, f1** — metriche di valutazione calcolate per ciascuna classe;
- **support** — numero di istanze appartenenti alla classe considerata;
- **alpha, beta** — soglie di rifiuto e accettazione utilizzate nel modello 3WD;
- **defer_ratio** — proporzione delle istanze per le quali il modello ha scelto di non prendere una decisione automatica;
- **class** — classe di riferimento della riga di risultati (0 per negativa, 1 per positiva);
- **dataset** — nome del dataset su cui è stata condotta la sperimentazione.

4.3.2 Visualizzazioni e grafici

Oltre ai report numerici, il sistema genera automaticamente una serie di grafici di supporto:

- **Grafico Coverage–Accuracy:** rappresenta la relazione tra tasso di copertura (1 - astensione) e accuratezza, consentendo di valutare l'impatto dell'astensione sulle prestazioni complessive;
- **Curve ROC positive e negative:** mostrano la capacità del modello di distinguere correttamente le classi, con evidenziazione dei punti corrispondenti alle soglie α e β ;
- **Diagrammi di calibrazione:** confrontano le probabilità predette con la percentuale reale di casi positivi, valutando la bontà della calibrazione dei modelli;
- **Istogrammi delle probabilità:** mostrano la distribuzione delle probabilità predette per ciascun modello;

- **Matrici di confusione:** rappresentano le prestazioni dei modelli sia nel caso binario sia nelle configurazioni 3WD;
- **Curve Precision–Recall:** analizzano il tradeoff tra precisione e recall calcolate su tutti i valori (anche quelli deferred). Sono evidenziati i punti corrispondenti alle soglie α e β .

4.4 Architettura del Software

4.4.1 Pipeline di esecuzione

L'intero processo, dalla preparazione dei dati alla valutazione dei modelli, è organizzato in una pipeline composta da diverse fasi principali:

1. Caricamento del dataset.

- Per il dataset *Bank Marketing*, i dati vengono recuperati direttamente dal repository UCI tramite la libreria *ucimlrepo*.
- Per il dataset *Adult Census Income*, i dati vengono scaricati dal sito UCI, puliti e preprocessati. In particolare, la colonna *fnlwgt* (peso di rappresentatività dell'individuo nel campione) viene rimossa in quanto non utile ai fini predittivi. Successivamente, viene creato un sottoinsieme bilanciato di 14.000 osservazioni (7.000 per ciascuna classe) per poter addestrare i modelli su un dataset bilanciato.

2. Preprocessamento e codifica. Le variabili categoriali vengono convertite in rappresentazioni numeriche mediante *one-hot encoding*.

3. Suddivisione del dataset. Ogni dataset viene suddiviso in *training set* e *test set*, contenenti rispettivamente il 70% e il 30% dei dati totali.

4. Addestramento e valutazione. Durante la fase di addestramento vengono esplorate diverse coppie di soglie (α, β) per valutare il comportamento del modello in presenza di regioni di astensione di dimensione diversa. Ogni modello viene addestrato sul *training set* e valutato sul *test set*. Le metriche calcolate durante la fase di test includono *accuracy*, *precision*, *recall*, *F1-score* e *defer_ratio*.

5. Esportazione dei risultati. I risultati sperimentali vengono poi salvati nel file CSV ed i vari grafici, descritti in precedenza, vengono generati.

4.4.2 Struttura del codice

Per svolgere questo processo sono stati creati due script principali:

- **main.py:** script principale che gestisce l'intera pipeline, dall'importazione dei dati alla valutazione dei modelli. [A.1]
- **functions.py:** contiene le funzioni di supporto per il caricamento dei dati, il preprocessamento, l'addestramento dei modelli e la generazione dei report. [A.2]

Capitolo 5

Risultati Sperimentali

In questo capitolo sono riportati i risultati sperimentali ottenuti applicando i metodi di classificazione considerati al dataset *Bank Marketing* e al dataset *Adult Census Income*.

5.1 Bank Marketing

5.1.1 Metriche principali

Le Tabelle 5.1 e 5.2 riportano le principali metriche di valutazione dei modelli per la classe positiva e negativa sul dataset *Bank Marketing*, estratte dal file CSV generato durante l'esecuzione del programma.

Modelli binari

Entrambi i modelli presentano un'accuratezza complessiva di circa 0.9, ma tale valore è fortemente influenzato dal predominio della classe negativa.

Per la **classe positiva**, si osservano valori di Precision moderati (0.66 per RF e 0.60 per MLP) e di Recall piuttosto bassi (0.38 e 0.46 rispettivamente), a indicare che i modelli individuano solo una parte dei veri positivi.

Al contrario, per la **classe negativa**, le metriche raggiungono valori molto elevati (Precision e Recall superiori a 0.92), segno di una forte tendenza del classificatore verso la classe maggioritaria, in coerenza con la natura sbilanciata del dataset *Bank Marketing*.

Modelli Three-Way Decision (3WD)

Al variare delle soglie α e β si modifica il bilanciamento tra le varie metriche e la copertura decisionale come segue:

- diminuendo α e aumentando β , il modello diventa più cauto e tende a rimandare un numero maggiore di istanze;
- il Deferred ratio cresce, così come la Precision, mentre il Recall diminuisce.

Nel caso del modello RF_3WD, passando da $(\alpha, \beta) = (0.3, 0.7)$ a $(0.1, 0.9)$, la Precision cresce drasticamente da 0.79 a 1.0, a discapito del Deferred ratio (aumento da 0.13 a 0.29) e del Recall (riduzione da 0.28 a 0.03). Questo conferma che soglie più restrittive portano a decisioni più affidabili, ma riducendo la capacità del modello di individuare tutti i positivi.

Comportamenti analoghi si riscontrano nei modelli MLP_3WD, che mostrano tuttavia una maggiore stabilità: la riduzione del Recall è meno drastica e il punteggio F1 rimane più equilibrato (da 0.53 a 0.30), segno che la rete neurale gestisce meglio la zona di indecisione.

Tabella 5.1: Metriche classe positiva – Bank Marketing

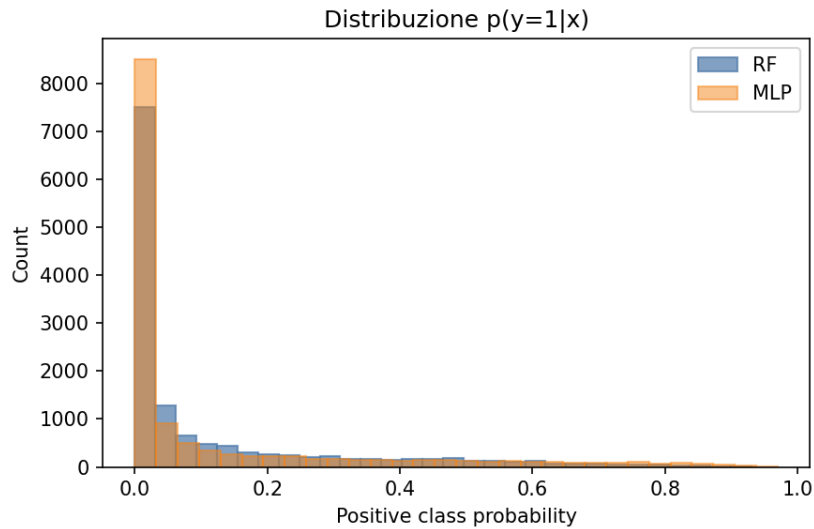
Model	Accuracy	Precision	Recall	F1	Support	α	β	Deferred ratio
RF_binary	0.9	0.66	0.38	0.48	1587	-	-	0
RF_3WD	0.95	0.79	0.28	0.42	710	0.3	0.7	0.13
RF_3WD	0.97	0.83	0.17	0.29	378	0.2	0.8	0.2
RF_3WD	0.99	1.0	0.03	0.07	146	0.1	0.9	0.29
MLP_binary	0.9	0.6	0.46	0.52	1587	-	-	0
MLP_3WD	0.94	0.7	0.43	0.53	890	0.3	0.7	0.12
MLP_3WD	0.96	0.79	0.37	0.51	544	0.2	0.8	0.18
MLP_3WD	0.98	0.83	0.18	0.3	241	0.1	0.9	0.26

Tabella 5.2: Metriche classe negativa – Bank Marketing

Model	Accuracy	Precision	Recall	F1	Support	α	β	Deferred ratio
RF_binary	0.9	0.92	0.97	0.95	11977	-	-	0
RF_3WD	0.95	0.96	1.0	0.98	11104	0.3	0.7	0.13
RF_3WD	0.97	0.97	1.0	0.98	10505	0.2	0.8	0.2
RF_3WD	0.99	0.99	1.0	0.99	9497	0.1	0.9	0.29
MLP_binary	0.9	0.93	0.96	0.94	11977	-	-	0
MLP_3WD	0.94	0.96	0.99	0.97	11086	0.3	0.7	0.12
MLP_3WD	0.96	0.97	0.99	0.98	10518	0.2	0.8	0.18
MLP_3WD	0.98	0.98	1.0	0.99	9767	0.1	0.9	0.26

5.1.2 Istogramma delle probabilità

Nella Figura 5.1 la distribuzione risulta fortemente asimmetrica, con la quasi totalità delle istanze concentrate verso lo 0, rendendo molto difficile la predizione dei casi appartenenti alla classe positiva. In questo scenario, il parametro β risulta fondamentale per la classificazione corretta delle istanze positive.

**Figura 5.1:** Istogramma delle probabilità predette per RF e MLP - Bank Marketing.

5.1.3 Matrici di confusione

Nella classificazione binaria, sia Random Forest che MLP evidenziano una significativa presenza di falsi negativi, riflettendo la difficoltà nel riconoscere correttamente la classe positiva, soprattutto in questo contesto nettamente sbilanciato. L'introduzione del 3WD consente di modulare il comportamento del classificatore: incrementando il valore di β e riducendo α , si restringe la zona di decisione, aumentando la cautela del modello. Questo comporta una diminuzione del numero di istanze classificate.

Il modello Random Forest tende a ottenere metriche leggermente superiori rispetto a MLP, sebbene ciò comporti un tasso di deferimento più elevato.

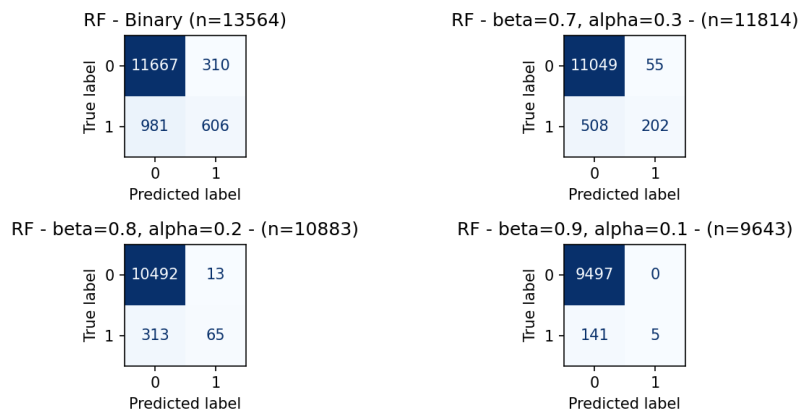


Figura 5.2: Matrici di confusione per Random Forest (binaria e 3WD) - Bank Marketing.

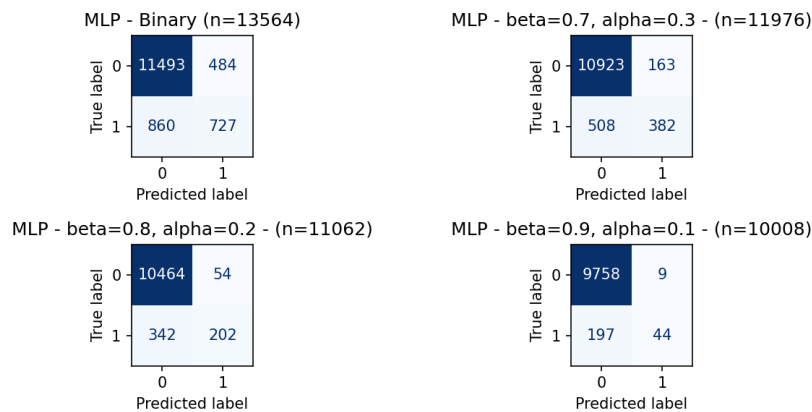


Figura 5.3: Matrici di confusione per MLP (binaria e 3WD) - Bank Marketing.

5.1.4 Curve ROC

Entrambi i modelli raggiungono prestazioni eccellenti con $AUC = 0.915$ e $AUC = 0.920$ per MLP e RF rispettivamente (Figure 5.4 e 5.5).

Per la classe negativa, i punti operativi di α si collocano nella regione ad alto TPR e basso FPR, indicando un'ottima capacità discriminativa. Per la classe positiva, i punti operativi di β mostrano un comportamento conservativo con bassi valori di TPR e FPR, riflettendo la difficoltà nel classificare correttamente la classe minoritaria.

Il Random Forest risulta essere leggermente migliore in entrambe le classi.

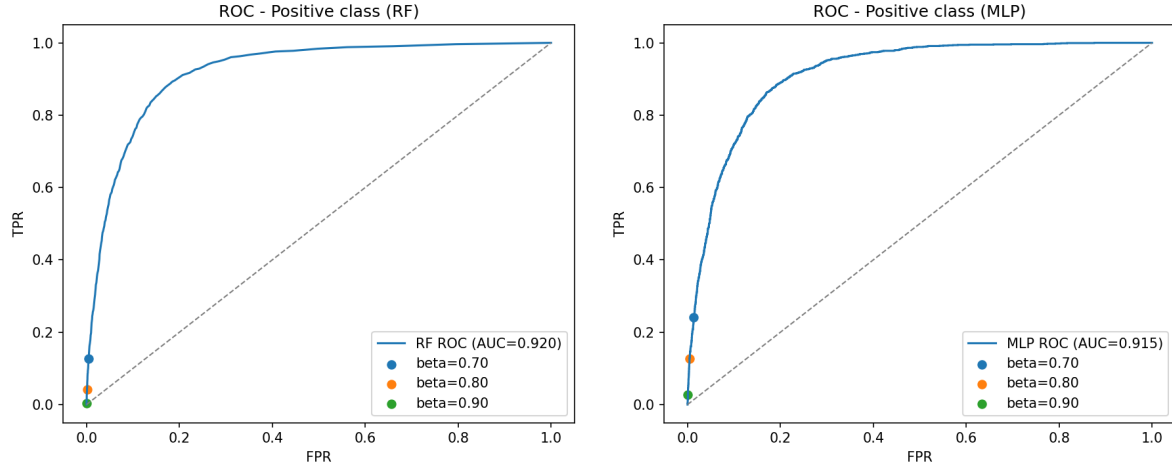


Figura 5.4: ROC (analisi positiva) per RF e MLP - Bank Marketing.

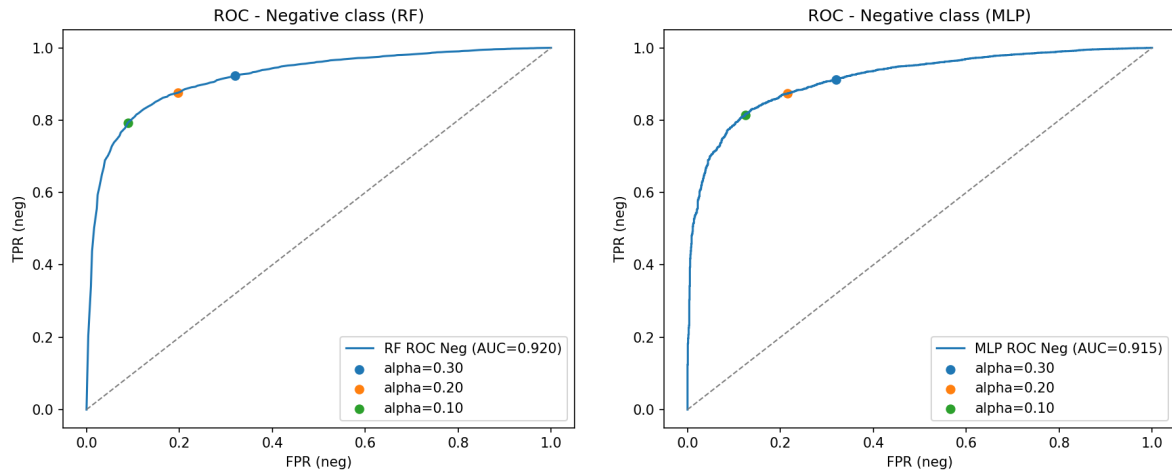


Figura 5.5: ROC (analisi negativa) per RF e MLP - Bank Marketing.

5.1.5 Diagrammi di Calibrazione

Nella configurazione binaria, in entrambi i modelli le curve si discostano leggermente dalla diagonale di calibrazione perfetta (Brier Score: RF=0.065, MLP=0.068). L'approccio three-way decision permette comunque di migliorare la calibrazione: all'aumentare di β e al diminuire di α , il Brier Score diminuisce progressivamente, raggiungendo valori inferiori a 0.020 per entrambi i modelli.

Nelle curve relative alla 3WD sono presenti meno punti rispetto alla configurazione binaria in quanto le istanze nella regione di deferimento non vengono classificate, riducendo il numero di bins utilizzati nel diagramma.

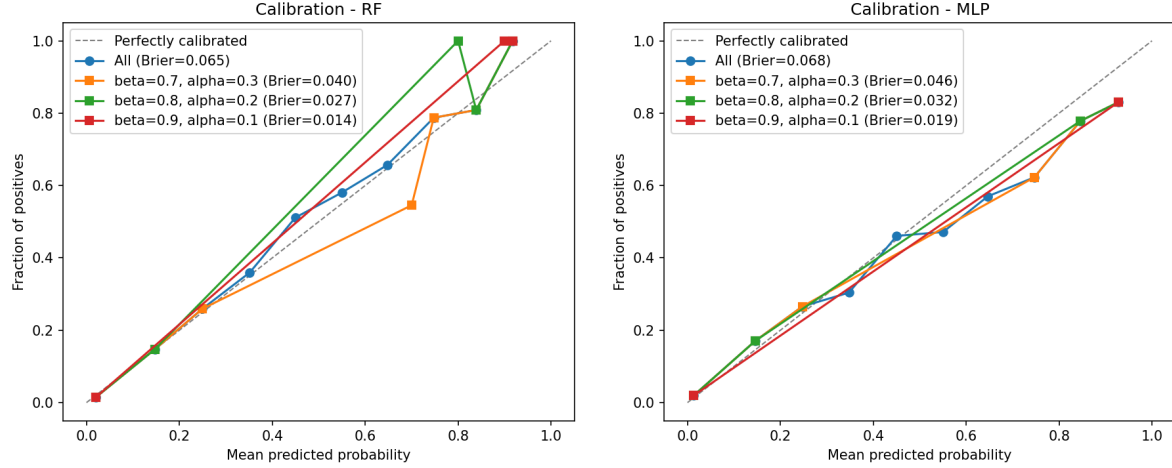


Figura 5.6: Diagrammi di calibrazione per RF e MLP - Bank Marketing.

5.1.6 Grafici Coverage-Accuracy

Entrambi i modelli dimostrano che riducendo la coverage si ottiene un incremento dell'accuracy, ma con una relazione fortemente non lineare: piccole variazioni di accuracy (1-2%) corrispondono a riduzioni drastiche della coverage (fino al 30-35%). Questo suggerisce che i valori ottimali di β e α siano rispettivamente vicini a 0.8 e 0.2, i quali permettono di mantenere un'accuracy molto elevata (95-98%), senza andare a compromettere drasticamente la coverage (75-85%).

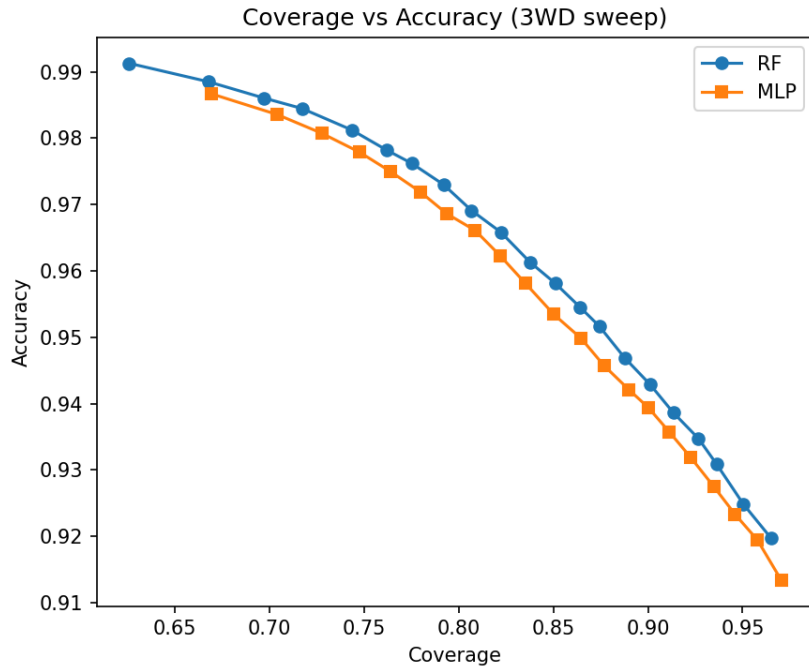


Figura 5.7: Coverage vs Accuracy (sweep simmetrico $\alpha=1-\beta$) - Bank Marketing.

5.1.7 Grafici Precision-Recall

Entrambi i modelli mostrano una forte asimmetria: la AP positiva risulta moderata (RF=0.600, MLP=0.577), mentre la AP negativa è quasi perfetta (0.987 per entrambi).

Per la classe negativa, gli OP di α restano nella zona ad altissima precision con lieve perdita di recall al diminuire di α , confermando la prevalenza dei casi negativi. Per la classe positiva, gli OP di β evidenziano come i modelli siano conservativi, mantenendo alta la precision e drasticamente basso il recall.

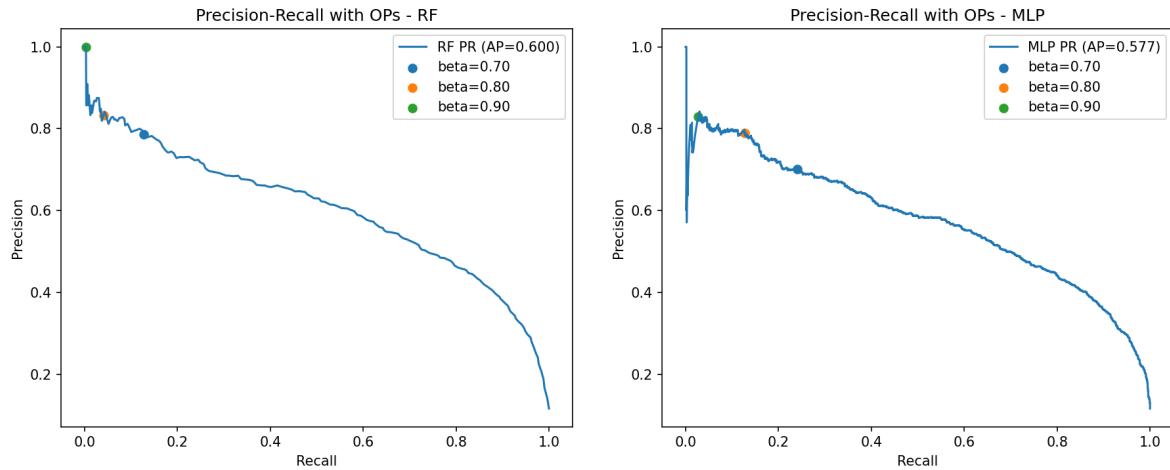


Figura 5.8: Curve Precision-Recall (analisi positiva) per RF e MLP - Bank Marketing.

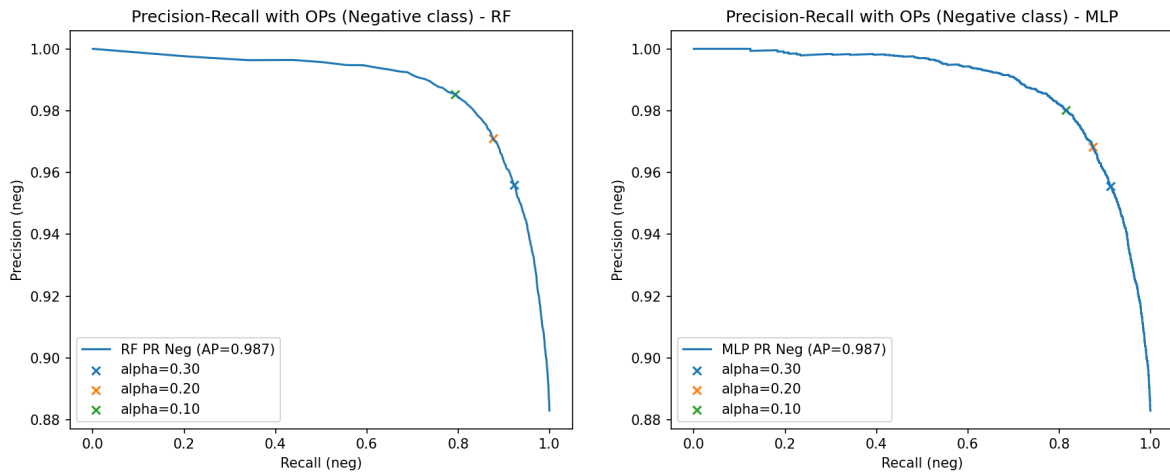


Figura 5.9: Curve Precision-Recall (analisi negativa) per RF e MLP - Bank Marketing.

5.2 Adult Census Income

5.2.1 Metriche principali

Le Tabelle 5.3 e 5.4 riportano le principali metriche di valutazione dei modelli per la classe positiva e negativa sul dataset *Adult Census Income*, estratte dal file CSV generato durante l'esecuzione del programma.

Modelli binari

I modelli binari (RF_binary e MLP_binary) mostrano prestazioni complessivamente buone e simmetriche tra le due classi, con valori medi di Accuracy, Precision, Recall e F1-score tra 0.80 e 0.84, in linea con le aspettative di un modello addestrato su un dataset bilanciato.

Modelli Three-Way Decision (3WD)

Al diminuire di α e all'aumentare di β , la seguente tendenza può essere osservata:

- il modello diventa più *conservativo*, rifiutando di classificare istanze con probabilità intermedie;
- a differenza del caso sbilanciato, il Recall aumenta;
- il deferred ratio cresce in modo significativo, fino a superare il 50% per soglie molto restrittive.

L'aumento del valore del Recall è dato dal fatto che ampliare la zona di deferral comporta l'esclusione principalmente delle istanze più ambigue con probabilità intermedie. Di conseguenza, diminuiscono sia i falsi positivi sia i falsi negativi, e il Recall calcolato sulle istanze effettivamente classificate tende ad aumentare.

Tabella 5.3: Metriche classe positiva – Adult Census Income

Model	Accuracy	Precision	Recall	F1	Support	α	β	Deferred ratio
RF_binary	0.82	0.82	0.82	0.82	2100	-	-	0
RF_3WD	0.89	0.88	0.89	0.88	1568	0.3	0.7	0.23
RF_3WD	0.92	0.9	0.93	0.91	1293	0.2	0.8	0.35
RF_3WD	0.94	0.92	0.96	0.94	933	0.1	0.9	0.51
MLP_binary	0.83	0.81	0.85	0.83	2100	-	-	0
MLP_3WD	0.89	0.88	0.91	0.89	1614	0.3	0.7	0.22
MLP_3WD	0.92	0.91	0.93	0.92	1294	0.2	0.8	0.35
MLP_3WD	0.96	0.96	0.95	0.95	878	0.1	0.9	0.52

Tabella 5.4: Metriche classe negativa – Adult Census Income

Model	Accuracy	Precision	Recall	F1	Support	α	β	Deferred ratio
RF_binary	0.82	0.82	0.82	0.82	2100	-	-	0
RF_3WD	0.89	0.9	0.88	0.89	1665	0.3	0.7	0.23
RF_3WD	0.92	0.93	0.91	0.92	1417	0.2	0.8	0.35
RF_3WD	0.94	0.97	0.93	0.95	1131	0.1	0.9	0.51
MLP_binary	0.83	0.84	0.8	0.82	2100	-	-	0
MLP_3WD	0.89	0.9	0.88	0.89	1648	0.3	0.7	0.22
MLP_3WD	0.92	0.93	0.91	0.92	1430	0.2	0.8	0.35
MLP_3WD	0.96	0.96	0.97	0.96	1148	0.1	0.9	0.52

5.2.2 Istogramma delle probabilità

Tramite la Figura 5.10 è possibile osservare una distribuzione bimodale, con una alta concentrazione marcata sia verso 0 che verso 1. Ciò indica che entrambi i modelli tendono

a predire la classe di appartenenza con elevata certezza, assegnando alle istanze valori di confidenza elevati.

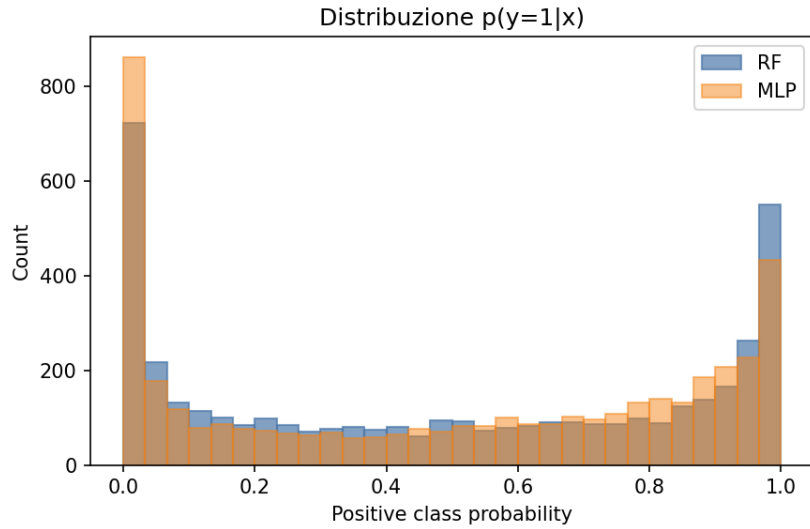


Figura 5.10: Istogramma delle probabilità predette per RF e MLP - Adult Census Income.

5.2.3 Matrici di confusione

Nella classificazione binaria, entrambi i modelli ottengono prestazioni equilibrate tra le due classi (in linea con il bilanciamento del dataset). Con l'approccio three-way decision, all'aumentare di β e al diminuire di α , si osserva:

- una riduzione progressiva relativamente elevata delle istanze classificate;
- una progressiva variazione nella distribuzione delle classi predette: il modello tende a classificare un maggior numero di TN, mentre il numero di TP diminuisce.
- un miglioramento complessivo delle metriche.

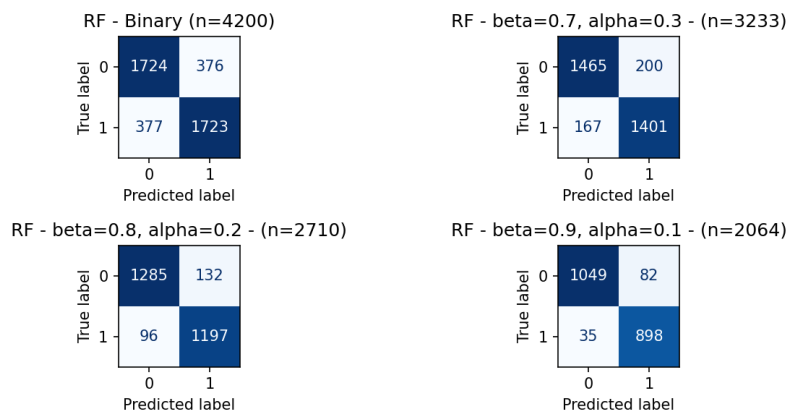


Figura 5.11: Matrici di confusione per Random Forest (binaria e 3WD) - Adult Census Income.

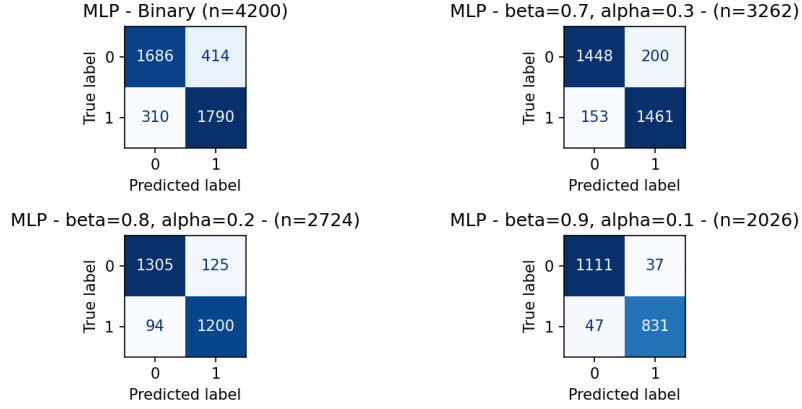


Figura 5.12: Matrici di confusione per MLP (binaria e 3WD) - Adult Census Income.

5.2.4 Curve ROC

Le Figure 5.13 e 5.14 mostrano prestazioni eccellenti per entrambi i classificatori ($AUC = 0.909$ per MLP, $AUC = 0.900$ per RF). Rispetto al dataset sbilanciato, si osserva un leggero decremento dell' AUC per entrambi i modelli, con il Random Forest che mostra una riduzione più marcata, risultando leggermente peggiore di MLP nel caso bilanciato.

Per la classe positiva, i punti operativi di β mostrano ora valori di TPR significativamente più elevati, indicando una migliore capacità di identificare correttamente i casi positivi quando le classi sono bilanciate. I punti operativi della classe negativa hanno prestazioni più basse del caso sbilanciato, ma comunque accettabili, riflettendo la robustezza data da un dataset bilanciato.

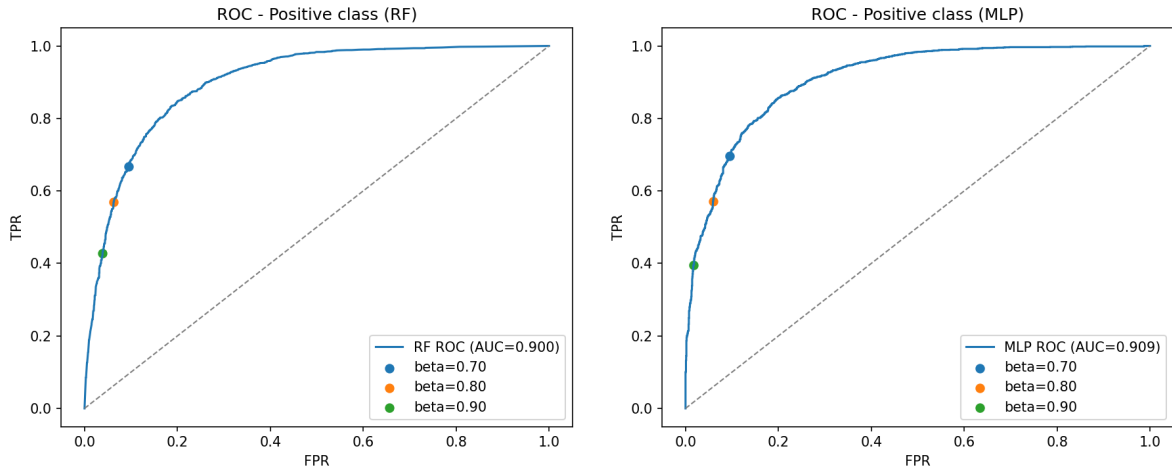


Figura 5.13: ROC (analisi positiva) per RF e MLP - Adult Census Income.

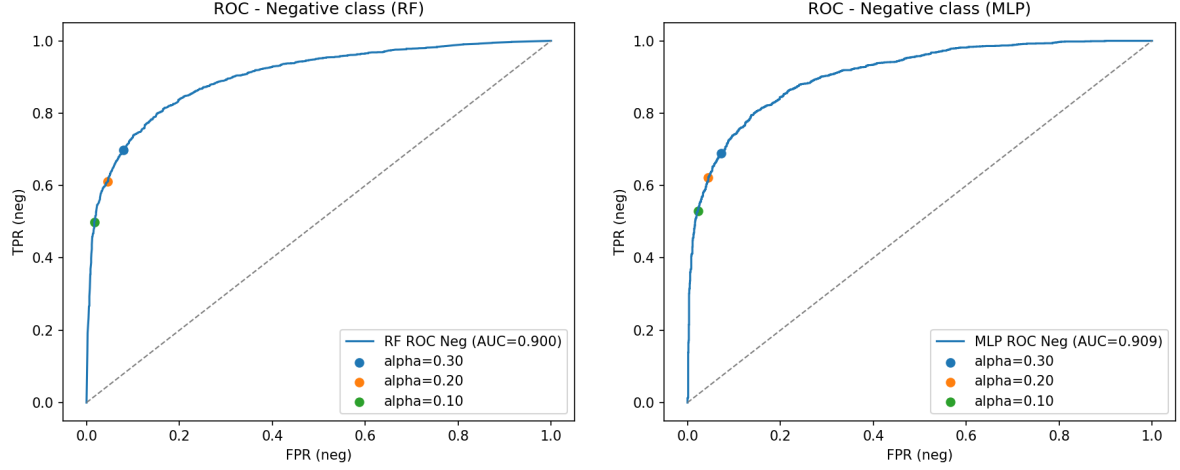


Figura 5.14: ROC (analisi negativa) per RF e MLP - Adult Census Income.

5.2.5 Diagrammi di Calibrazione

Rispetto al dataset sbilanciato, entrambi i modelli presentano una calibrazione peggiore nella configurazione binaria. La 3WD migliora la calibrazione, con Brier Score che diminuisce progressivamente all'aumentare di β e al diminuire di α , raggiungendo valori di 0.054 (RF) e 0.039 (MLP), ottenendo comunque una calibrazione inferiore rispetto al dataset sbilanciato. In questo caso, però, la maggior parte dei punti delle linee si trovano sopra la diagonale, indicando una tendenza del modello a sottostimare le probabilità degli eventi.

E' riconfermato che L'MLP possiede prestazioni migliori del RF sul dataset bilanciato, come già osservato in precedenza.

Allo stesso modo, le curve con three-way decision presentano meno punti a causa del deferimento delle istanze.

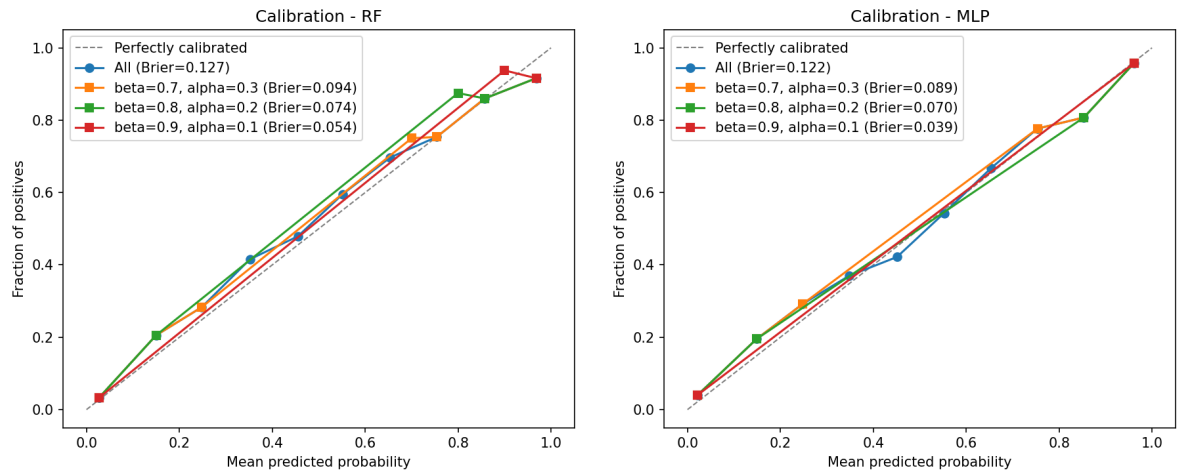


Figura 5.15: Diagrammi di calibrazione per RF e MLP - Adult Census Income.

5.2.6 Grafici Coverage-Accuracy

Anche qui si osserva una relazione fortemente non lineare: piccole riduzioni di accuracy permettono di incrementare notevolmente la coverage, ma in modo meno marcato del caso sbilanciato.

Anche in questo caso, valori ottimali potrebbero essere vicini a 0.2 e 0.8 per α e β rispettivamente, garantendo coverage del 60-70% ed accuracy del 92-94%.

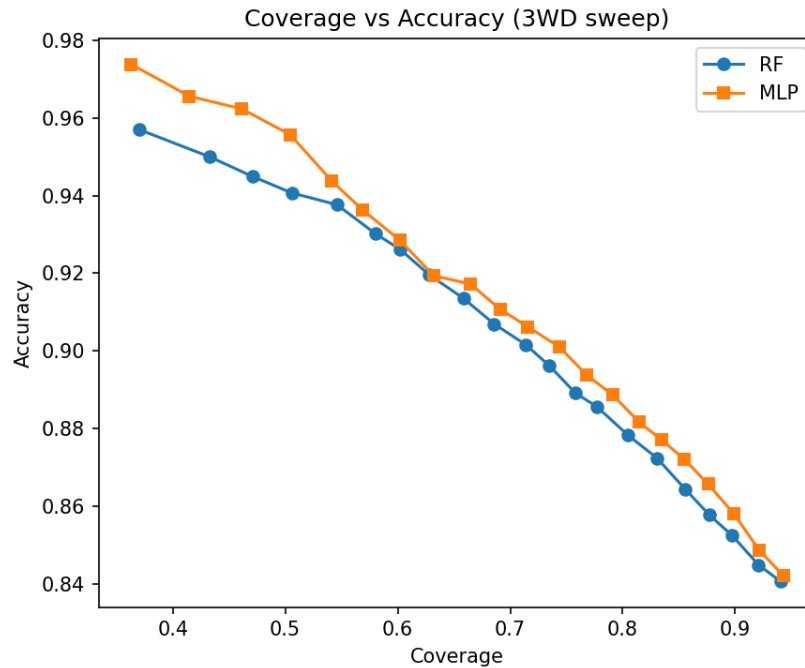


Figura 5.16: Coverage vs Accuracy (sweep simmetrico $\alpha=1-\beta$) - Adult Census Income.

5.2.7 Grafici Precision-Recall

In questo caso, l'asimmetria vista nel caso sbilanciato scompare: l'AP della classe positiva cresce nettamente (RF 0.600 \rightarrow 0.884; MLP 0.577 \rightarrow 0.904) mentre quella della negativa cala (0.987 \rightarrow RF 0.908, MLP 0.912).

I valori ottimali di α e β potrebbero essere compresi tra 0.1-0.2 e 0.8-0.9 rispettivamente, in quanto diminuendo leggermente la precisione (85-90%), è possibile ottenere valori relativamente alti di recall (55-65%) per entrambe le classi.

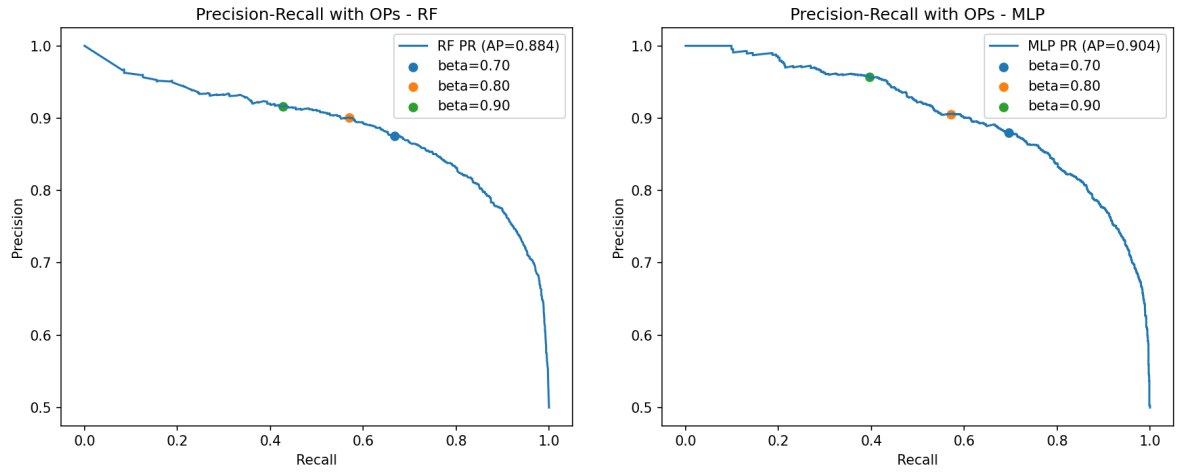


Figura 5.17: Curve Precision-Recall (analisi positiva) per RF e MLP - Adult Census Income.

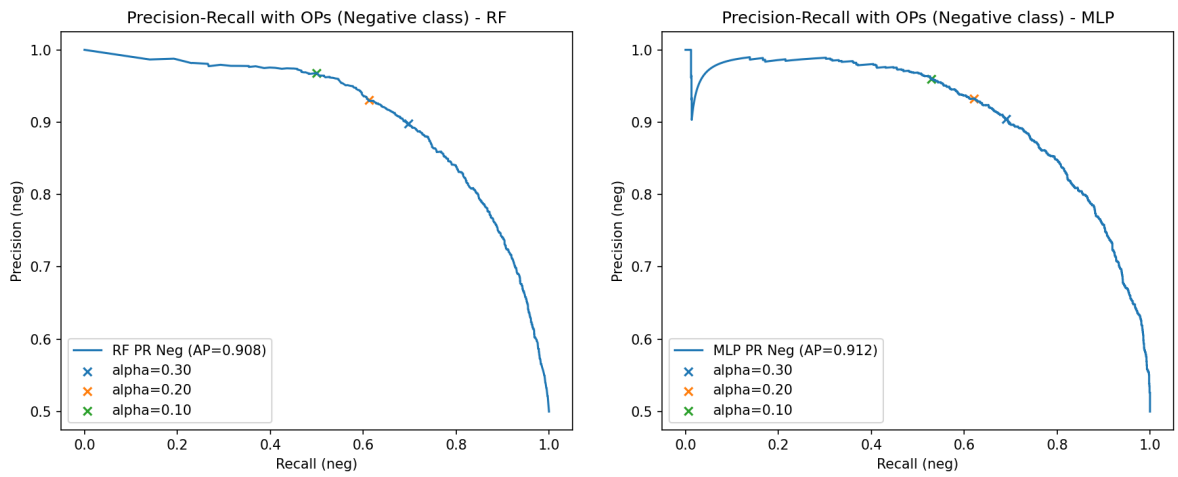


Figura 5.18: Curve Precision-Recall (analisi negativa) per RF e MLP - Adult Census Income.

Capitolo 6

Conclusione e Sviluppi Futuri

In questo progetto si è studiato l'applicazione dell'approccio Three Way Decision (3WD) a problemi di classificazione binaria, confrontando due dataset con caratteristiche diverse, *Bank Marketing* (caso sbilanciato) e *Adult Census Income* (caso bilanciato), tramite due classificatori diversi, *Random Forest* (RF) e *Multi-Layer Perceptron* (MLP).

E' stata implementata una pipeline di esecuzione, che include:

- **Preprocessing:** sono state effettuate pulizia, normalizzazione, standardizzazione e preparazione dei dataset.
- **Training e Valutazione:** i due classificatori sono stati addestrati e valutati sia in configurazione binaria che in configurazione 3WD.
- **Analisi dei Risultati:** per ogni dataset è stato generato un report in formato CSV, accompagnato da vari grafici di supporto con metriche avanzate.

6.1 Sintesi dei Risultati

In base a quanto visto con il dataset *Bank Marketing*, l'approccio 3WD permette di migliorare nettamente le prestazioni, mantenendo sempre un tasso di deferimento relativamente basso. Nei contesti applicativi dove un errore può comportare rischi elevati, i modelli 3WD risultano particolarmente vantaggiosi: essi migliorano le prestazioni, soprattutto per la classe in minoranza, e riducono il rischio di errore.

Nel caso del dataset bilanciato *Adult Census Income*, l'applicazione dell'approccio 3WD ha portato a un miglioramento significativo delle prestazioni. Tuttavia, questo risultato è stato ottenuto con un tasso di deferimento più elevato rispetto al caso sbilanciato, limitando i vantaggi pratici dell'approccio in questo contesto.

Per entrambi i dataset, i valori ottimali delle soglie α e β sono risultati essere intorno a 0.1 – 0.2 e 0.8 – 0.9 rispettivamente, permettendo di avere un buon compromesso tra prestazioni dei modelli e tasso di astensione.

Nel complesso, i risultati confermano che 3WD è una strategia utile quando si vuole aumentare l'affidabilità delle decisioni e si ha la possibilità di trattare le istanze deferite con un processo secondario.

6.2 Sviluppi futuri

Di seguito sono proposti alcuni sviluppi concreti per proseguire ed ampliare il lavoro:

- **Integrazione di strategie per gestire i deferimenti:** implementare un secondo livello decisionale (es. revisione umana o modello specialistico) e valutare il costo complessivo del processo aggiuntivo.
- **Ottimizzazione automatica delle soglie:** usare tecniche di ricerca automatica ed ottimizzazione per trovare coppie (α, β) che massimizzino una funzione di utilità basata sulle metriche principali e sul costo di eventuali errori.
- **Analisi più approfondita di α e β :** in questo studio è stato eseguito un semplice sweep simmetrico per valutare l'impatto delle soglie. Un'analisi più approfondita potrebbe esplorare una griglia più ampia di valori, anche non simmetrici, per identificare configurazioni ottimali in base allo scenario applicativo.
- **Valutazione su più dataset e classificatori:** estendere lo studio ad altri dataset con caratteristiche diverse e includere altri modelli di classificazione.
- **Studio degli iperparametri dei modelli:** in questo studio non sono stati considerati gli effetti degli iperparametri dei modelli, che potrebbero comunque influenzare significativamente le prestazioni.

Appendice A

Codice Python

A.1 main.py

```
1 from sklearn.ensemble import RandomForestClassifier
2 from sklearn.model_selection import train_test_split
3 from sklearn.metrics import accuracy_score, classification_report
4 from sklearn.neural_network import MLPClassifier
5 from sklearn.preprocessing import StandardScaler
6 import numpy as np
7 import os
8 import matplotlib
9
10 from utils.functions import (
11     prepare_dataset, plot_negative_roc, plot_positive_roc,
12     plot_calibration, plot_confusion_matrices, plot_histogram,
13     plot_coverage_accuracy, pr_positive_with_beta,
14     pr_negative_with_alpha, init_classification_metrics_csv,
15     save_classification_metrics
16 )
17
18 datasets = ["bank", "adult"] # Options: "bank" or/and "adult"
19
20 matplotlib.use('Agg')
21
22 for dataset in datasets:
23     print(f"\n{'=' * 80}")
24     print(f"PROCESSING DATASET: {dataset.upper()}")
25     print(f"{'=' * 80}")
26
27     X, y, dataset_name = prepare_dataset(dataset)
28
29     X_train, X_test, y_train, y_test = train_test_split(
30         X, y, test_size=0.3, random_state=42, stratify=y
31     )
32
33     beta_points = [0.70, 0.80, 0.90]
34     alpha_points = [0.30, 0.20, 0.10]
35
36     dataset_suffix = dataset_name.lower().replace(" ", "_")
37     RUN_DIR = os.path.join("outputs", f"{dataset_suffix}")
38     csv_report_path = os.path.join(RUN_DIR, f"{dataset_suffix}
39         _report.csv")
40     os.makedirs(RUN_DIR, exist_ok=True)
```

```

38 os.makedirs(os.path.join(RUN_DIR, "pr_3wd"), exist_ok=True)
39 os.makedirs(os.path.join(RUN_DIR, "roc_3wd"), exist_ok=True)
40
41 init_classification_metrics_csv(csv_report_path)
42
43 # =====
44 # Random Forest
45 # =====
46 print(f"\n>> Training Random Forest on {dataset_name} dataset...
      ")
47
48 clf = RandomForestClassifier(n_estimators=200, random_state=42,
49                             n_jobs=-1)
49 clf.fit(X_train, y_train)
50
51 # Binary prediction
52 y_pred_binary = clf.predict(X_test)
53
54 accuracy_binary = accuracy_score(y_test, y_pred_binary)
55 class_report_binary = classification_report(y_test,
56                                             y_pred_binary, output_dict=True)
56 save_classification_metrics(class_report_binary, "RF_binary",
57                             accuracy_binary, None, None, 0, dataset_name,
58                             csv_report_path)
59
60 # 3WD prediction
61 probs = clf.predict_proba(X_test)[: , 1]
62
63 for (beta, alpha) in zip(beta_points, alpha_points):
64     decisions = np.where(probs >= beta, 1, np.where(probs <=
65             alpha, 0, -1))
66
67     accepted_indices = decisions != -1
68     y_true_certain = y_test[accepted_indices]
69     y_pred_certain = decisions[accepted_indices]
70
71     accuracy_certain = accuracy_score(y_true_certain,
72                                     y_pred_certain)
73     deferred_ratio = np.mean(decisions == -1)
74     class_report_certain = classification_report(y_true_certain,
75                                                 y_pred_certain, output_dict=True)
76     save_classification_metrics(class_report_certain, "RF_3WD",
77                             accuracy_certain, alpha, beta, deferred_ratio,
78                             dataset_name, csv_report_path)
79
80 print(">> Random Forest Training and Evaluation Completed.")
81
82 # =====
83 # Multi Layer Perceptron
84 # =====

```

```

81 print(f"\n>> Training Neural Network (MLP) on {dataset_name}
    dataset...")
82
83 # Feature scaling for MLP
84 scaler = StandardScaler()
85 X_train_nn = scaler.fit_transform(X_train)
86 X_test_nn = scaler.transform(X_test)
87
88 mlp = MLPClassifier(
89     hidden_layer_sizes=(128, 64),
90     activation='relu',
91     solver='adam',
92     learning_rate='adaptive',
93     max_iter=200,
94     random_state=42,
95     early_stopping=True,
96     n_iter_no_change=10,
97 )
98 mlp.fit(X_train_nn, y_train)
99
100 # Binary prediction
101 y_pred_binary_nn = mlp.predict(X_test_nn)
102
103 accuracy_binary_nn = accuracy_score(y_test, y_pred_binary_nn)
104 class_report_binary_nn = classification_report(y_test,
105     y_pred_binary_nn, output_dict=True)
106 save_classification_metrics(class_report_binary_nn, "MLP_binary"
107     , accuracy_binary_nn, None, None, 0, dataset_name,
108     csv_report_path)
109
110 # 3WD prediction
111 probs_nn = mlp.predict_proba(X_test_nn)[: , 1]
112
113 for (beta, alpha) in zip(beta_points, alpha_points):
114     decisions_nn = np.where(probs_nn >= beta, 1, np.where(
115         probs_nn <= alpha, 0, -1))
116
117     accepted_indices_nn = decisions_nn != -1
118     y_true_certain_nn = y_test[accepted_indices_nn]
119     y_pred_certain_nn = decisions_nn[accepted_indices_nn]
120
121     accuracy_certain_nn = accuracy_score(y_true_certain_nn,
122         y_pred_certain_nn)
123     deferred_ratio_nn = np.mean(decisions_nn == -1)
124     class_report_certain_nn = classification_report(
125         y_true_certain_nn, y_pred_certain_nn, output_dict=True)
126     save_classification_metrics(class_report_certain_nn, "
127         MLP_3WD", accuracy_certain_nn, alpha, beta,
128         deferred_ratio_nn, dataset_name,
129         csv_report_path)

```

```

124 print(">> Neural Network Training and Evaluation Completed.")
125
126 # =====
127 # 1) Coverage-Accuracy sweep (RF e MLP)
128 # =====
129
130 # symmetric sweep: alpha = 1 - beta
131 betas_cov_acc = np.linspace(0.55, 0.95, 21)
132 alphas_cov_acc = 1.0 - betas_cov_acc
133
134 plot_coverage_accuracy(y_test, probs, probs_nn, betas_cov_acc,
135                        alphas_cov_acc,
136                        os.path.join(RUN_DIR, "
137                                     coverage_accuracy_rf_mlp.png"))
138
139 # =====
140 # 2) ROC/PR
141 # =====
142
143 os.makedirs(os.path.join(RUN_DIR, "roc_3wd"), exist_ok=True)
144 plot_positive_roc(y_test, probs, "RF", os.path.join(RUN_DIR, "
145               roc_3wd", "rf_roc_positive.png"),
146               beta_points=beta_points)
147 plot_positive_roc(y_test, probs_nn, "MLP", os.path.join(RUN_DIR,
148               "roc_3wd", "mlp_roc_positive.png"),
149               beta_points=beta_points)
150 plot_negative_roc(y_test, probs, "RF", os.path.join(RUN_DIR, "
151               roc_3wd", "rf_roc_negative.png"),
152               alpha_points=alpha_points)
153 plot_negative_roc(y_test, probs_nn, "MLP", os.path.join(RUN_DIR,
154               "roc_3wd", "mlp_roc_negative.png"),
155               alpha_points=alpha_points)
156
157 # =====
158 # 3) Calibration diagrams + Brier
159 # =====
160
161 mask_rf_certain = decisions != -1
162 plot_calibration(y_test, probs, beta_points, alpha_points, "RF",
163                os.path.join(RUN_DIR, "rf_calibration.png"))
164
165 mask_mlp_certain = decisions_nn != -1
166 plot_calibration(y_test, probs_nn, beta_points, alpha_points, "
167               MLP", os.path.join(RUN_DIR, "mlp_calibration.png"))
168
169 # =====
170 # 4) Probability histograms
171 # =====
172
173 plot_histogram(probs, probs_nn, "Distribuzione p(y=1|x)", os.
174               path.join(RUN_DIR, "prob_hist.png"))

```

```

166
167 # =====
168 # 5) Confusion matrices
169 # =====
170
171 plot_confusion_matrices(y_test, y_pred_binary, probs,
172                        beta_points, alpha_points, "RF",
173                        os.path.join(RUN_DIR, "
174                                     rf_confusion_matrices.png"))
175
176 plot_confusion_matrices(y_test, y_pred_binary_nn, probs_nn,
177                        beta_points, alpha_points, "MLP",
178                        os.path.join(RUN_DIR, "
179                                     mlp_confusion_matrices.png"))
180
181 # =====
182 # 6) Precision-Recall curves
183 # =====
184
185 pr_positive_with_beta(
186     y_test.values if hasattr(y_test, 'values') else y_test,
187     probs,
188     "RF",
189     beta_points,
190     os.path.join(RUN_DIR, "pr_3wd"),
191 )
192 pr_positive_with_beta(
193     y_test.values if hasattr(y_test, 'values') else y_test,
194     probs_nn,
195     "MLP",
196     beta_points,
197     os.path.join(RUN_DIR, "pr_3wd"),
198 )
199 pr_negative_with_alpha(
200     y_test.values if hasattr(y_test, 'values') else y_test,
201     probs,
202     "RF",
203     alpha_points,
204     os.path.join(RUN_DIR, "pr_3wd"),
205 )
206 pr_negative_with_alpha(
207     y_test.values if hasattr(y_test, 'values') else y_test,
208     probs_nn,
209     "MLP",
210     alpha_points,
211     os.path.join(RUN_DIR, "pr_3wd"),
212 )
213
214 # =====
215 # Utility: show outputs folder
216 # =====

```



```

213     print(f"\n{'=' * 60}")
214     print(f"THREE-WAY DECISION ANALYSIS - {dataset_name.upper()}")
215     print(f"{'=' * 60}")
216     print(f"Output directory: {RUN_DIR}")
217     print(f"Training samples: {len(X_train)}")
218     print(f"Test samples: {len(X_test)}")
219     print(f"Features: {X.shape[1]}")
220     print(f"Target distribution: {y.value_counts().to_dict()}")
221     print(f"{'=' * 60}\n")
222
223     print(f"\n{'=' * 80}")
224     print("ALL DATASETS PROCESSED SUCCESSFULLY!")
225     print("Check the 'outputs/' directory for results")
226     print(f"{'=' * 80}")

```

A.2 functions.py

```

1
2 import os
3 import numpy as np
4 import pandas as pd
5 from matplotlib import pyplot as plt
6 from sklearn.calibration import calibration_curve
7 from sklearn.metrics import roc_curve, roc_auc_score,
8     brier_score_loss, ConfusionMatrixDisplay, confusion_matrix, \
9     accuracy_score, precision_recall_curve, average_precision_score
10 from sklearn.utils import resample
11 from ucimlrepo import fetch_ucirepo
12 import csv
13
14 #####
15 # Dataset loading functions
16 #####
17
18
19 def load_bank_marketing_dataset():
20     print("Loading Bank Marketing dataset...")
21     bank_marketing = fetch_ucirepo(id=222)
22
23     X = bank_marketing.data.features
24     y = bank_marketing.data.targets
25
26     if isinstance(y, pd.DataFrame) and y.shape[1] == 1:
27         y = y.iloc[:, 0]
28
29     if y.dtype == object or str(y.dtype).startswith('category'):
30         y_mapped = y.map({'yes': 1, 'no': 0})
31         if not y_mapped.isna().any():
32             y = y_mapped.astype(int)

```

```

33
34     return X, y, "Bank Marketing"
35
36
37 def load_adult_census_dataset():
38     print("Loading Adult Census Income dataset...")
39
40     url = 'https://archive.ics.uci.edu/ml/machine-learning-databases
         /adult/adult.data'
41
42     column_names = [
43         'age', 'workclass', 'fnlwgt', 'education', 'education-num',
44         'marital-status',
45         'occupation', 'relationship', 'race', 'sex', 'capital-gain',
46         'capital-loss',
47         'hours-per-week', 'native-country', 'income'
48     ]
49
50     try:
51         df = pd.read_csv(url, names=column_names, na_values='?',
52                          skipinitialspace=True)
53         print(f"Loaded {len(df)} samples from Adult Census Income
54               dataset")
55     except Exception as e:
56         print(f"Error loading dataset from URL: {e}")
57         print("Trying alternative approach...")
58         # Alternative: try to load from local file if URL fails
59         return None, None, None
60
61     df_clean = df.dropna()
62     print(f"After removing missing values: {len(df_clean)} samples")
63
64     df_clean.drop(columns=['fnlwgt'], inplace=True)
65     print("Dropped 'fnlwgt' column")
66
67     # Balance the dataset by creating a subset of 14k samples (7k
68     # per class)
69     print("Creating balanced subset...")
70     df_majority = df_clean[df_clean['income'] == '<=50K']
71     df_minority = df_clean[df_clean['income'] == '>50K']
72
73     n_samples_per_class = 7000
74     df_majority_downsampled = resample(
75         df_majority,
76         replace=False,
77         n_samples=n_samples_per_class,
78         random_state=42
79     )
80     df_minority_downsampled = resample(
81         df_minority,
82         replace=False,

```

```

78         n_samples=n_samples_per_class,
79         random_state=42
80     )
81     df_balanced = pd.concat([df_majority_downsampled,
82                             df_minority_downsampled])
83     df_balanced = df_balanced.sample(frac=1, random_state=42).
84         reset_index(drop=True)
85
86     X = df_balanced.drop('income', axis=1)
87     y = df_balanced['income']
88
89     # Map target to binary: '<=50K' -> 0, '>50K' -> 1
90     y = y.map({'<=50K': 0, '>50K': 1}).astype(int)
91
92     return X, y, "Adult Census Income"
93
94 def prepare_dataset(dataset_name):
95     if dataset_name.lower() == "bank":
96         X, y, name = load_bank_marketing_dataset()
97     elif dataset_name.lower() == "adult":
98         X, y, name = load_adult_census_dataset()
99     else:
100         raise ValueError("Dataset must be 'bank' or 'adult'")
101
102     if X is None or y is None:
103         raise ValueError(f"Failed to load {dataset_name} dataset")
104
105     # One-hot encoding for categorical variables
106     X = pd.get_dummies(X, drop_first=True)
107
108     print(f"Final feature matrix shape: {X.shape}")
109     print(f"Target distribution: {y.value_counts().to_dict()}")
110
111     return X, y, name
112
113 # =====
114 # CSV report functions
115 # =====
116
117 def init_classification_metrics_csv(csv_path):
118     with open(csv_path, mode='w', newline='') as f:
119         writer = csv.writer(f)
120         writer.writerow(['model', 'accuracy', 'precision', 'recall',
121                         'f1', 'support', 'alpha', 'beta', 'defer_ratio',
122                         'class', 'dataset'])
123
124 def save_classification_metrics(report_dict, model, accuracy, alpha,
125                                beta, defer, dataset, csv_path):

```

```

125     with open(csv_path, mode='a', newline='') as f:
126         writer = csv.writer(f)
127         for cls in ['0', '1']:
128             precision = round(report_dict[cls]['precision'], 2)
129             recall = round(report_dict[cls]['recall'], 2)
130             f1 = round(report_dict[cls]['f1-score'], 2)
131             support = int(report_dict[cls]['support'])
132             row = [model, round(accuracy, 2), precision, recall, f1,
133                   support, alpha, beta, round(defer, 2), int(cls),
134                   dataset]
135             writer.writerow(row)
136
137 # =====
138 # ROC/PR functions
139 # =====
140
141 def plot_positive_roc(y_true, prob_values, model_name, out_path,
142                      beta_points):
143     fpr, tpr, _ = roc_curve(y_true, prob_values)
144     auc = roc_auc_score(y_true, prob_values)
145
146     plt.figure(figsize=(6, 5))
147     plt.plot(fpr, tpr, label=f"{model_name} ROC (AUC={auc:.3f})")
148     plt.plot([0, 1], [0, 1], "--", color="gray", linewidth=1)
149
150     for a in beta_points:
151         y_pred_a = (prob_values >= a).astype(int)
152         tp_a = np.sum((y_true == 1) & (y_pred_a == 1))
153         fp_a = np.sum((y_true == 0) & (y_pred_a == 1))
154         tn_a = np.sum((y_true == 0) & (y_pred_a == 0))
155         fn_a = np.sum((y_true == 1) & (y_pred_a == 0))
156         fpr_a = fp_a / (fp_a + tn_a) if (fp_a + tn_a) > 0 else 0.0
157         tpr_a = tp_a / (tp_a + fn_a) if (tp_a + fn_a) > 0 else 0.0
158         plt.scatter([fpr_a], [tpr_a], label=f"beta={a:.2f}", s=35)
159
160     plt.xlabel("FPR")
161     plt.ylabel("TPR")
162     plt.title(f"ROC - Positive class ({model_name})")
163     plt.legend()
164     plt.tight_layout()
165     plt.savefig(out_path, dpi=150)
166     plt.close()
167
168 def plot_negative_roc(y_true, prob_values, model_name, out_path,
169                      alpha_points):
170     y_arr = y_true.values if hasattr(y_true, 'values') else y_true
171     y_neg = (1 - y_arr).astype(int)
172     p_neg = 1.0 - prob_values
173
174     fpr_neg, tpr_neg, _ = roc_curve(y_neg, p_neg)

```

```

173 auc_neg = roc_auc_score(y_neg, p_neg)
174
175 plt.figure(figsize=(6, 5))
176 plt.plot(fpr_neg, tpr_neg, label=f"{model_name} ROC Neg (AUC={
    auc_neg:.3f})")
177
178 for b in alpha_points:
179     thr_b = 1.0 - b
180     y_pred_b = (p_neg >= thr_b).astype(int)
181     tp_b = np.sum((y_neg == 1) & (y_pred_b == 1))
182     fp_b = np.sum((y_neg == 0) & (y_pred_b == 1))
183     tn_b = np.sum((y_neg == 0) & (y_pred_b == 0))
184     fn_b = np.sum((y_neg == 1) & (y_pred_b == 0))
185     fpr_b = fp_b / (fp_b + tn_b) if (fp_b + tn_b) > 0 else 0.0
186     tpr_b = tp_b / (tp_b + fn_b) if (tp_b + fn_b) > 0 else 0.0
187     plt.scatter([fpr_b], [tpr_b], label=f"alpha={b:.2f}")
188
189 plt.plot([0, 1], [0, 1], "--", color="gray", linewidth=1)
190 plt.xlabel("FPR (neg)")
191 plt.ylabel("TPR (neg)")
192 plt.title(f"ROC - Negative class ({model_name})")
193 plt.legend()
194 plt.tight_layout()
195 plt.savefig(out_path, dpi=150)
196 plt.close()
197
198
199 # =====
200 # Calibration functions
201 # =====
202 def plot_calibration(y_true, prob_values, betas, alphas, title,
    out_name):
203     plt.figure(figsize=(6, 5))
204
205     frac_pos_all, mean_pred_all = calibration_curve(y_true,
        prob_values, n_bins=10, strategy="uniform")
206     brier_all = brier_score_loss(y_true, prob_values)
207
208     plt.plot([0, 1], [0, 1], "--", color="gray", linewidth=1, label=
        "Perfectly calibrated")
209     plt.plot(mean_pred_all, frac_pos_all, marker="o", label=f"All (
        Brier={brier_all:.3f})")
210
211     for (a, b) in zip(betas, alphas):
212         decisions = np.where(prob_values >= a, 1, np.where(
            prob_values <= b, 0, -1))
213         decisions_mask = decisions != -1
214
215         y_certain = y_true[decisions_mask]
216         p_certain = prob_values[decisions_mask]
217         frac_pos_certain, mean_pred_certain = calibration_curve(

```

```

218         y_certain,
219         p_certain,
220         n_bins=min(10, max(2, int(np.sqrt(len(p_certain))))),
221         strategy="uniform"
222     )
223     brier_certain = brier_score_loss(y_certain, p_certain)
224
225     plt.plot(mean_pred_certain, frac_pos_certain,
226              marker="s", label=f"beta={a}, alpha={b} (Brier={
227                  brier_certain:.3f})")
228
229     plt.xlabel("Mean predicted probability")
230     plt.ylabel("Fraction of positives")
231     plt.title(f"Calibration - {title}")
232     plt.legend()
233     plt.tight_layout()
234     plt.savefig(out_name, dpi=150)
235     plt.close()
236
237 # =====
238 # Confusion matrix functions
239 # =====
240
241 def plot_confusion_matrices(y_true, y_pred_binary_model, probs,
242                             betas, alphas, title, out_name):
243     fig, axes = plt.subplots(2, 2, figsize=(10, 4))
244     ConfusionMatrixDisplay(
245         confusion_matrix(y_true, y_pred_binary_model, labels=[0, 1])
246         ,
247         display_labels=[0, 1]
248     ).plot(ax=axes[0][0], colorbar=False, cmap="Blues",
249           values_format="d")
250     n_all = len(y_true)
251     axes[0][0].set_title(f"{title} - Binary (n={n_all})")
252
253     for i, (a, b) in enumerate(zip(betas, alphas), 1):
254         decisions = np.where(probs >= a, 1, np.where(probs <= b, 0,
255             -1))
256         accepted_indices = decisions != -1
257         y_true_certain = y_true[accepted_indices]
258         y_pred_certain = decisions[accepted_indices]
259
260         ConfusionMatrixDisplay(
261             confusion_matrix(y_true_certain, y_pred_certain, labels
262                             =[0, 1]),
263             display_labels=[0, 1]
264         ).plot(ax=axes[i // 2][i % 2], colorbar=False, cmap="Blues",
265               values_format="d")
266         n_certain = len(y_true_certain)

```

```

261         axes[i // 2][i % 2].set_title(f"{title} - beta={a}, alpha={b  

262             } - (n={n_certain})")
263
264     plt.tight_layout()
265     plt.savefig(out_name, dpi=150)
266     plt.close()
267
268     # =====
269     # Histogram functions
270     # =====
271     def plot_histogram(prob_rf, prob_mlp, title, out_name):
272         plt.figure(figsize=(6, 4))
273         plt.hist(prob_rf, bins=30, color="#4C78A8", alpha=0.7, label="RF  

274             ", edgecolor="#4C78A8")
275         plt.hist(prob_mlp, bins=30, color="#F58518", alpha=0.5, label="
276             MLP", edgecolor="#F58518")
277         plt.xlabel("Positive class probability")
278         plt.ylabel("Count")
279         plt.title(f"{title}")
280         plt.legend()
281         plt.tight_layout()
282         plt.savefig(out_name, dpi=150)
283         plt.close()
284
285     # =====
286     # Coverage Accuracy functions
287     # =====
288     def coverage_accuracy_curve(y_true, prob_values, betas, alphas):
289         coverages = []
290         accuracies = []
291         for a, b in zip(betas, alphas):
292             decisions_tmp = np.where(prob_values >= a, 1, np.where(
293                 prob_values <= b, 0, -1))
294             mask = decisions_tmp != -1
295             coverage = np.mean(mask)
296             if coverage > 0:
297                 acc = accuracy_score(y_true[mask], decisions_tmp[mask])
298             else:
299                 acc = np.nan
300             coverages.append(coverage)
301             accuracies.append(acc)
302         return np.array(coverages), np.array(accuracies)
303
304     def plot_coverage_accuracy(y_test, probs_rf, probs_mlp, betas,
305         alphas, out_name):
306         cov_rf, acc_rf = coverage_accuracy_curve(y_test, probs_rf, betas
307             , alphas)
308         cov_mlp, acc_mlp = coverage_accuracy_curve(y_test, probs_mlp,
309             betas, alphas)

```

```

305 plt.figure(figsize=(6, 5))
306 plt.plot(cov_rf, acc_rf, marker="o", label="RF")
307 plt.plot(cov_mlp, acc_mlp, marker="s", label="MLP")
308 plt.xlabel("Coverage")
309 plt.ylabel("Accuracy")
310 plt.title(f"Coverage vs Accuracy (3WD sweep)")
311 plt.legend()
312 plt.tight_layout()
313 plt.savefig(out_name, dpi=150)
314 plt.close()
315
316 # =====
317 # Precision-Recall functions
318 # =====
319 def pr_positive_with_beta(y_true, prob_values, model_name,
320 beta_points, out_dir):
321     os.makedirs(out_dir, exist_ok=True)
322     prec, rec, _ = precision_recall_curve(y_true, prob_values)
323     ap = average_precision_score(y_true, prob_values)
324
325     plt.figure(figsize=(6, 5))
326     plt.plot(rec, prec, label=f"{model_name} PR (AP={ap:.3f})")
327
328     for a in beta_points:
329         y_pred_beta = (prob_values >= a).astype(int)
330         tp = np.sum((y_true == 1) & (y_pred_beta == 1))
331         fp = np.sum((y_true == 0) & (y_pred_beta == 1))
332         fn = np.sum((y_true == 1) & (y_pred_beta == 0))
333         prec_beta = tp / (tp + fp) if (tp + fp) > 0 else 0.0
334         rec_beta = tp / (tp + fn) if (tp + fn) > 0 else 0.0
335         plt.scatter([rec_beta], [prec_beta], label=f"beta={a:.2f}",
336                     marker="o")
337
338     plt.xlabel("Recall")
339     plt.ylabel("Precision")
340     plt.title(f"Precision-Recall with OPs - {model_name}")
341     plt.legend()
342     plt.tight_layout()
343     out_path = os.path.join(out_dir, f"{model_name.lower()}
344         _pr_multi_3wd.png")
345     plt.savefig(out_path, dpi=150)
346     plt.close()
347
348 def pr_negative_with_alpha(y_true, prob_values, model_name,
349 alpha_points, out_dir):
350     os.makedirs(out_dir, exist_ok=True)
351     y_neg = 1 - y_true
352     p_neg = 1 - prob_values
353     prec, rec, _ = precision_recall_curve(y_neg, p_neg)
354     ap = average_precision_score(y_neg, p_neg)

```



```

352
353 plt.figure(figsize=(6, 5))
354 plt.plot(rec, prec, label=f"{model_name} PR Neg (AP={ap:.3f})")
355
356 for b in alpha_points:
357     thr_neg = 1.0 - b
358     y_pred_thr = (p_neg >= thr_neg).astype(int)
359     tp = np.sum((y_neg == 1) & (y_pred_thr == 1))
360     fp = np.sum((y_neg == 0) & (y_pred_thr == 1))
361     fn = np.sum((y_neg == 1) & (y_pred_thr == 0))
362     prec_b = tp / (tp + fp) if (tp + fp) > 0 else 0.0
363     rec_b = tp / (tp + fn) if (tp + fn) > 0 else 0.0
364     plt.scatter([rec_b], [prec_b], label=f"alpha={b:.2f}",
365                 marker="x")
366
367 plt.xlabel("Recall (neg)")
368 plt.ylabel("Precision (neg)")
369 plt.title(f"Precision-Recall with OPs (Negative class) - {
370           model_name}")
371 plt.legend()
372 plt.tight_layout()
373 out_path = os.path.join(out_dir, f"{model_name.lower()}
374                           _pr_negative_alpha.png")
375 plt.savefig(out_path, dpi=150)
376 plt.close()

```

Bibliografia

- [1] Andrea Campagner, Riccardo Bellazzi et al. «Three-Way Classification: Ambiguity and Abstention in Machine Learning». In: *Knowledge-Based Systems* 163 (2019), pp. 358–370 (cit. alle pp. 3, 4).
- [2] Dheeru Dua e Casey Graff. *UCI Machine Learning Repository: Adult Data Set*. <https://archive.ics.uci.edu/ml/datasets/adult>. 2019 (cit. a p. 6).
- [3] Yue Gao. «Three-Way Decision Models in Probabilistic Rough Sets». Tesi di dott. University of Regina, 2016 (cit. a p. 2).
- [4] Xiaoyu Liu e Wen Liu. «Three-Way Fuzzy Decision Models for Uncertain Information». In: *IEEE Transactions on Fuzzy Systems* (2023). DOI: 10.1109/TFUZZ.2023.3250158 (cit. a p. 3).
- [5] Sérgio Moro, Paulo Cortez e Paulo Rita. «A Data-Driven Approach to Predict the Success of Bank Telemarketing». In: *Decision Support Systems* 62 (2014), pp. 22–31. DOI: 10.1016/j.dss.2014.03.001 (cit. a p. 5).
- [6] Niels Schreuder, Riccardo D. Prieto e Isabel Valera. «Fair Classification with Abstention». In: *Proceedings of the 38th International Conference on Machine Learning (ICML)*. 2021 (cit. a p. 4).
- [7] Jie Wang e Hui Zhang. «A Multi-Criteria Three-Way Decision Model for MADM Problems». In: *Symmetry* 14.4 (2022). DOI: 10.3390/sym14040812 (cit. a p. 3).
- [8] Yiyu Yao. «The Two Sides of the Theory of Three-Way Decisions». In: *Rough Sets and Knowledge Technology*. Vol. 6954. Lecture Notes in Computer Science. Springer, 2011, pp. 1–8 (cit. alle pp. 2, 3).
- [9] Yiyu Yao. «Three-Way Decisions with Probabilistic Rough Sets». In: *Information Sciences* 180.3 (2010), pp. 341–353. DOI: 10.1016/j.ins.2009.09.021 (cit. alle pp. 2, 3).
- [10] Yiyu Yao e Jingtao Qi. «Three-Way Decision Making Approach to Conflict Analysis and Resolution Using Probabilistic Rough Sets». In: *Information Sciences* 330 (2016), pp. 17–30 (cit. a p. 2).