Valutazione delle Prestazioni dei Sistemi di Machine Learning

Basato sulle slide di R. Basili, S. Filice
Adattato e ampliato da AI Assistant

Aprile 2025

Sommario

Questo documento esplora le metodologie e le metriche fondamentali per la valutazione delle prestazioni dei sistemi di Machine Learning (ML). Partendo dalla necessità di misurare l'efficacia di un modello, si analizzano le metriche più comuni per i classificatori, come la matrice di confusione, precision, recall e F1-score. Si discutono le sfide legate a dati sbilanciati e si introducono metriche aggregate come le medie (macro e micro) e il break-even point. Vengono inoltre presentate le corrette metodologie di valutazione, inclusa la suddivisione del dataset e la validazione incrociata. Infine, si affronta la diagnostica degli errori, concentrandosi sui concetti di Bias (Sottostima) e Varianza (Sovrastima) e su come interpretarli attraverso le curve di apprendimento per migliorare i modelli. Il documento include un caso di studio sulla classificazione di testi Reuters per illustrare l'applicazione pratica dei concetti.

Indice

1 Introduzione e Motivazioni

Nel campo del Machine Learning (ML), la costruzione di modelli predittivi o classificatori è solo una parte del lavoro. Una volta sviluppato un algoritmo o addestrato un modello, sorgono domande cruciali:

- Il sistema di ML sta funzionando correttamente rispetto agli obiettivi prefissati?
- Avendo a disposizione diversi algoritmi o configurazioni di modelli, quale offre le prestazioni migliori per uno specifico compito (task)?
- Quali azioni si possono intraprendere per migliorare le prestazioni di un sistema di classificazione esistente?

Rispondere a queste domande richiede un processo sistematico di **valutazione delle prestazioni**, basato su metriche quantitative e metodologie rigorose. Senza una valutazione adeguata, è impossibile confrontare oggettivamente diversi approcci, identificare i punti deboli di un sistema o misurare i progressi nello sviluppo.

Questo documento fornisce una panoramica delle principali metriche e tecniche utilizzate per valutare i sistemi di ML, con un focus particolare sui problemi di classificazione.

2 Metriche di Valutazione per Classificatori

La valutazione di un classificatore inizia spesso con l'analisi di come le sue previsioni si confrontano con i valori reali (ground truth) su un insieme di dati mai visto durante l'addestramento (test set).

2.1 Matrice di Confusione

La **matrice di confusione** è uno strumento fondamentale che riassume le prestazioni di un classificatore. Per un problema di classificazione con N classi, è una matrice $N \times N$ dove le righe rappresentano le classi reali (Actual) e le colonne rappresentano le classi predette (Predicted) dal modello.

L'elemento (i, j) della matrice indica il numero di istanze appartenenti alla classe reale i che sono state classificate dal modello come appartenenti alla classe j.

Esempio (3 classi): [source: 6]

		Valore Predetto		
		Classe A	Classe B	Classe C
Valore Reale	Classe A	38 (TP_A)	12 (FN_A, FP_B)	0 (FN_A, FP_C)
	Classe B	5 (FP_A, FN_B)	43 (TP_B)	2 (FN_B, FP_C)
	Classe C	6 (FP_A, FN_C)	0 (FP_B, FN_C)	44 (TP_C)

Gli elementi sulla **diagonale principale** (True Positives per ogni classe, TP) rappresentano le classificazioni corrette. Gli elementi fuori dalla diagonale rappresentano gli errori:

- **False Positives (FP)** per la classe j: istanze classificate come j ma appartenenti ad altre classi (somma sulla colonna j, esclusa la diagonale).
- **False Negatives (FN)** per la classe i: istanze appartenenti alla classe i ma classificate erroneamente in altre classi (somma sulla riga i, esclusa la diagonale).

Per una specifica classe C_i:

- TP_i: Numero di istanze di C_i classificate correttamente come C_i.
- FP_i: Numero di istanze non di C_i classificate erroneamente come C_i.
- FN_i: Numero di istanze di C_i classificate erroneamente come non C_i.
- TN_i: Numero di istanze non di C_i classificate correttamente come non C_i.

2.1.1 Accuracy e Error Rate

Dalla matrice di confusione si possono derivare metriche aggregate semplici:

- **Accuracy (Accuratezza):** La proporzione di classificazioni corrette sul totale. [source: 7]
$$\text{Accuracy} = \frac{\text{Numero di classificazioni corrette}}{\text{Numero totale di classificazioni}} = \frac{\sum_i TP_i}{\text{Totale istanze}} \quad (1)$$
Nell'esempio: Accuracy $= \frac{38+43+44}{150} = \frac{125}{150} \approx 83.33\%$.
- **Error Rate (Tasso di Errore):** La proporzione di classificazioni errate sul totale.
$$\text{Error Rate} = \frac{\text{Numero di classificazioni errate}}{\text{Numero totale di classificazioni}} = 1 - \text{Accuracy} \quad (2)$$
Nell'esempio: Error Rate $= \frac{12+5+2+6}{150} = \frac{25}{150} \approx 16.67\%$.

2.1.2 Limiti dell'Accuracy con Dati Sbilanciati

L'accuracy può essere **ingannevole** in presenza di classi sbilanciate (una o più classi sono molto più frequenti delle altre). [source: 8] Consideriamo un classificatore di spam, dove il 99.9% delle email non sono spam. Un classificatore banale che etichetta *tutte* le email come "Non-Spam" otterrebbe un'accuracy del 99.9%, pur fallendo completamente nell'identificare lo spam. [source: 10, 11]

		Valore Predetto	
		Spam	Non-Spam
Valore Reale	Spam	0	10
	Non-Spam	0	9990

Accuracy = (0 + 9990) / 10000 = 99.9%. [source: 11] Questo modello è inutile per il suo scopo (trovare spam), ma ha un'accuracy altissima. Ciò dimostra la necessità di metriche più specifiche per classe.

2.2 Metriche Basate su Classe Singola

Per superare i limiti dell'accuracy, si definiscono metriche focalizzate sulle prestazioni relative a ciascuna classe singolarmente, spesso considerando una classe "positiva" (quella di interesse) contro tutte le altre ("negative"). Le definizioni si basano sui concetti di True Positive (TP), False Positive (FP), True Negative (TN) e False Negative (FN) relativi alla classe di interesse C. [source: 13]

		Valore Predetto	
		Classe C	Non Classe C
Valore Reale	Classe C	TP	FN
	Non Classe C	FP	TN

Le metriche principali sono:

- **Precision (Precisione):** Misura la proporzione di istanze classificate come positive che sono effettivamente positive. Risponde alla domanda: "Di tutte le istanze che il classificatore ha etichettato come appartenenti alla classe C, quante ne fanno veramente parte?" [source: 14]

$$\text{Precision}_C = \frac{TP_C}{TP_C + FP_C} \tag{3}$$

  Una precisione alta indica che il classificatore commette pochi errori nell'assegnare la classe C (pochi falsi positivi).

- **Recall (Richiamo) o Sensitivity (Sensibilità) o True Positive Rate (TPR):** Misura la proporzione di istanze positive reali che sono state correttamente identificate dal classificatore. Risponde alla domanda: "Di tutte le istanze che appartengono veramente alla classe C, quante sono state identificate correttamente dal classificatore?" [source: 15]

$$\text{Recall}_C = \frac{TP_C}{TP_C + FN_C} \tag{4}$$

  Una recall alta indica che il classificatore è bravo a trovare tutte le istanze della classe C (pochi falsi negativi).

- **F1-Score (o F-measure):** È la **media armonica** di Precision e Recall. Fornisce un bilanciamento tra le due metriche, utile quando entrambe sono importanti. [source: 14, 15]

$$F1_C = \frac{2 \times \text{Precision}_C \times \text{Recall}_C}{\text{Precision}_C + \text{Recall}_C} = \frac{2 \times TP_C}{2 \times TP_C + FP_C + FN_C} \tag{5}$$

  La media armonica tende a dare più peso ai valori più bassi. Un F1-score alto si ottiene solo se sia la precisione sia la recall sono alte.

Esempio di Calcolo (dalla matrice precedente): [source: 22, 23, 30]

- **Classe A:**
  - $TP_A = 38$
  - $FP_A = 5 + 6 = 11$
  - $FN_A = 12 + 0 = 12$
  - $\text{Precision}_A = \frac{38}{38+11} = \frac{38}{49} \approx 0.775$
  - $\text{Recall}_A = \frac{38}{38+12} = \frac{38}{50} = 0.76$
  - $F1_A = \frac{2 \times 0.775 \times 0.76}{0.775+0.76} \approx 0.767$
- **Classe B:**
  - $TP_B = 43$
  - $FP_B = 12 + 0 = 12$
  - $FN_B = 5 + 2 = 7$
  - $\text{Precision}_B = \frac{43}{43+12} = \frac{43}{55} \approx 0.782$
  - $\text{Recall}_B = \frac{43}{43+7} = \frac{43}{50} = 0.86$
  - $F1_B = \frac{2 \times 0.782 \times 0.86}{0.782+0.86} \approx 0.819$
- **Classe C:**
  - $TP_C = 44$

- $FP_C = 0 + 2 = 2$
- $FN_C = 0 + 6 = 6$
- $\text{Precision}_C = \frac{44}{44+2} = \frac{44}{46} \approx 0.957$
- $\text{Recall}_C = \frac{44}{44+6} = \frac{44}{50} = 0.88$
- $F1_C = \frac{2 \times 0.957 \times 0.88}{0.957+0.88} \approx 0.917$

2.3 Compromesso Precision-Recall (Trade-off)

Spesso esiste un compromesso (trade-off) tra Precision e Recall: migliorare una può peggiorare l'altra. [source: 19] Molti classificatori producono uno score di confidenza e permettono di impostare una **soglia (threshold)**: le istanze con score superiore alla soglia vengono classificate come positive.

- **Soglia alta:** Il classificatore è molto cauto. Assegna la classe positiva solo alle istanze più sicure. Questo porta a pochi FP (alta Precision), ma potrebbe mancare molte istanze positive (bassa Recall).
- **Soglia bassa:** Il classificatore è più permissivo. Assegna la classe positiva a molte istanze. Questo cattura più istanze positive reali (alta Recall), ma aumenta il rischio di FP (bassa Precision).

Questo trade-off può essere visualizzato con una **curva Precision-Recall**, dove ogni punto rappresenta una diversa soglia. L'obiettivo ideale è il punto (1, 1) (massima recall e massima precision). [Image 1]

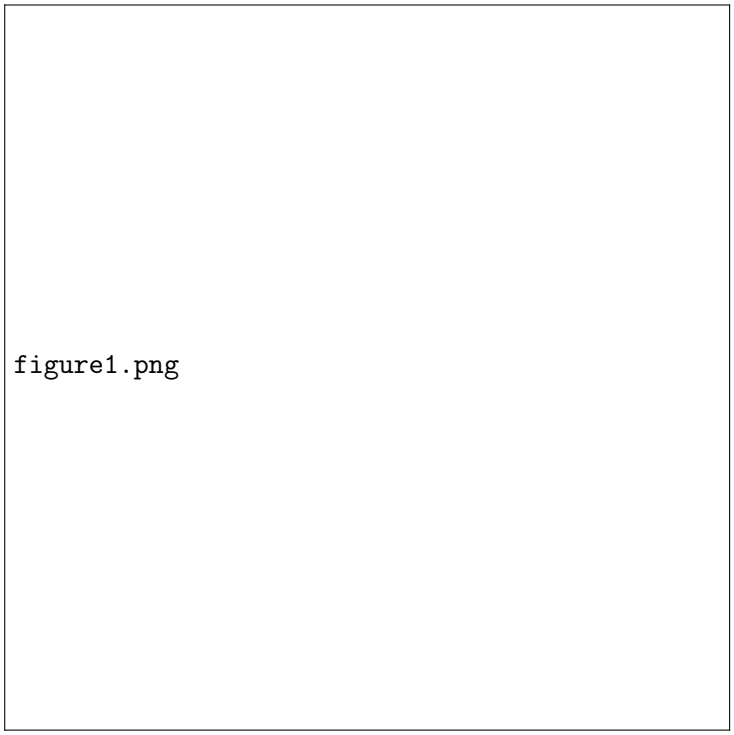

Figura 1: Illustrazione del compromesso Precision-Recall. Il punto ideale è (1,1).

3 Metriche Aggregate e Medie

Quando si lavora con più classi o si vuole un singolo numero per riassumere le prestazioni, si usano metriche aggregate o medie delle metriche per classe.

3.1 Break-even Point (BEP)

Il **Break-Even Point (BEP)** è un modo per riassumere la curva Precision-Recall. Rappresenta il valore (spesso interpolato) per cui Precision = Recall. [source: 24, 25] Un BEP più alto indica prestazioni migliori, poiché significa che si può ottenere un buon livello sia di precisione che di richiamo contemporaneamente. Viene spesso utilizzato per confrontare sistemi diversi sulla stessa curva P-R.

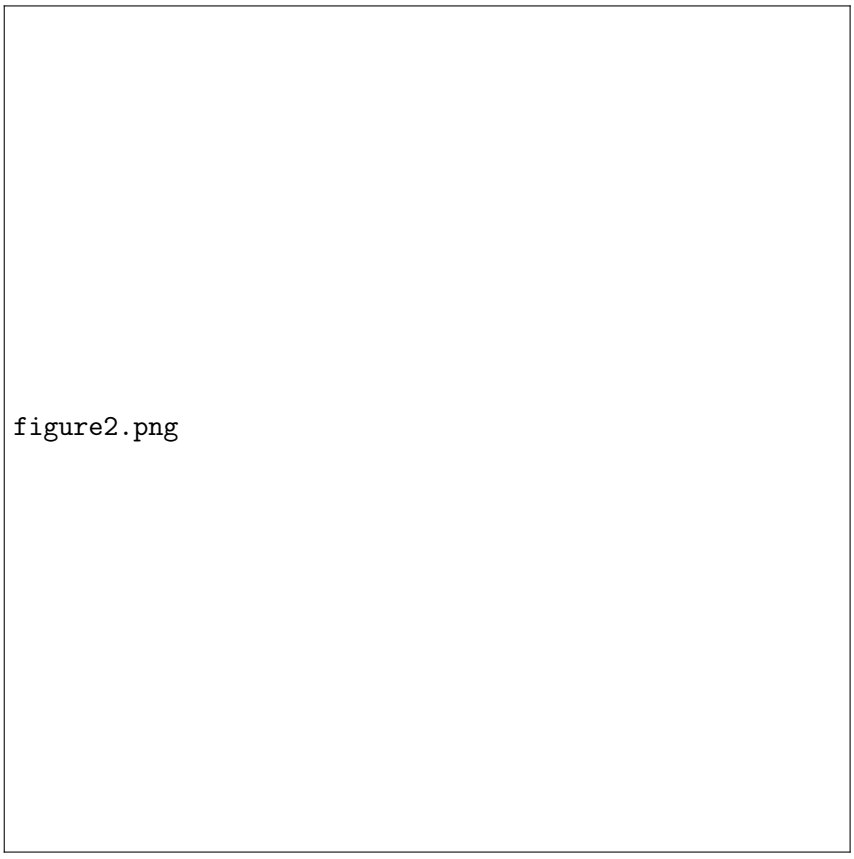

Figura 2: Esempio di curva Precision-Recall con indicazione del Break-Even Point (punto di intersezione con la bisettrice, P=R). Il BEP è più alto per il metodo "Stem".

3.2 F-measure come Media Armonica

Come già introdotto, l'F1-score è la media armonica di Precision (P) e Recall (R). Perché usare la media armonica invece di quella aritmetica o geometrica? [source: 26]

- Media Aritmetica: $\frac{P+R}{2}$
- Media Geometrica: $\sqrt{P \times R}$
- Media Armonica: $F1 = \frac{2}{\frac{1}{P}+\frac{1}{R}} = \frac{2PR}{P+R}$

La media armonica penalizza maggiormente i valori estremi. Se una tra Precision e Recall è molto bassa, l'F1-score sarà basso. La media aritmetica, invece, può dare un valore alto anche

se una delle due metriche è pessima (es. P=1, R=0.1 -> Media Aritm = 0.55; F1 = 0.18). L'F1 è quindi considerato un indicatore più robusto del bilanciamento tra P e R. [Image 3]

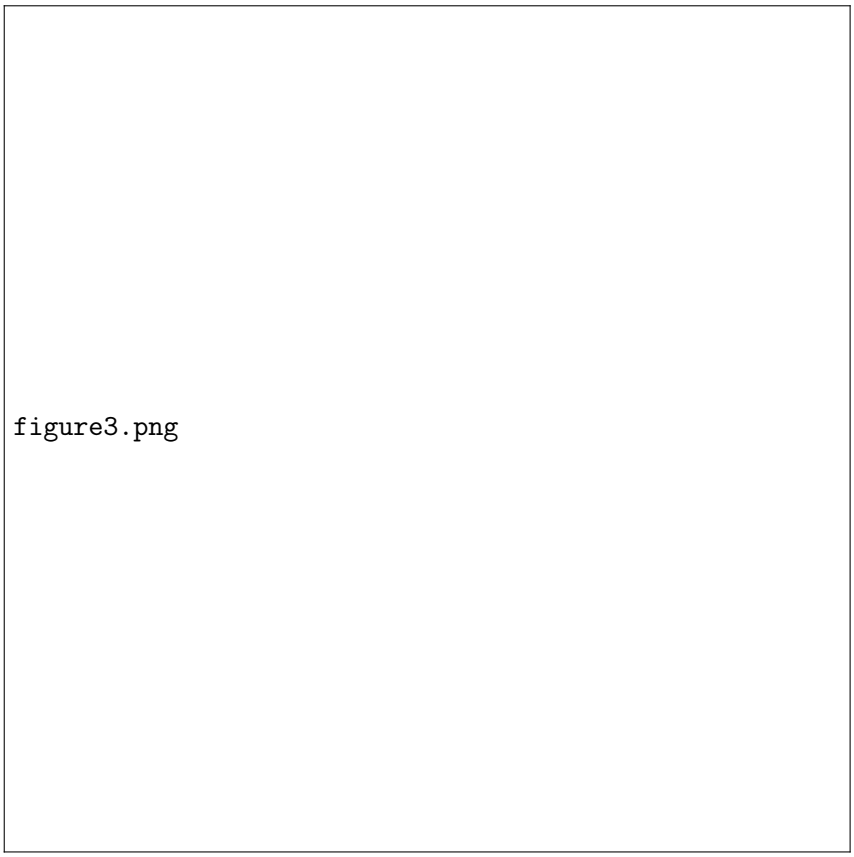

Figura 3: Confronto tra diverse medie di Precision e Recall (con Recall fissata al 70%). La media armonica (F1) è più sensibile a valori bassi di Precision.

3.3 Medie Cross-Categorical: Macro vs Micro

Per ottenere una valutazione complessiva su un problema multi-classe, si possono calcolare le medie delle metriche (Precision, Recall, F1) calcolate per ogni classe. Esistono due approcci principali: [source: 24, 27, 28]

- **Macro-averaging:** Si calcola la metrica (es. Precision) per ogni classe singolarmente e poi si fa la media aritmetica di questi valori.

$$\text{Macro-Precision} = \frac{1}{N}\sum_{i=1}^{N}\text{Precision}_i \tag{6}$$

$$\text{Macro-Recall} = \frac{1}{N}\sum_{i=1}^{N}\text{Recall}_i \tag{7}$$

$$\text{Macro-F1} = \frac{2 \times \text{Macro-Precision} \times \text{Macro-Recall}}{\text{Macro-Precision} + \text{Macro-Recall}} \tag{8}$$

  Questo approccio dà lo stesso peso a ogni classe, indipendentemente dalla sua dimensione. È utile se le prestazioni su classi rare sono importanti quanto quelle su classi comuni.

- **Micro-averaging:** Si aggregano i conteggi di TP, FP, FN per tutte le classi e poi si calcola la metrica globale una sola volta. Siano TP_i, FP_i, FN_i i conteggi per la classe i.

$$\text{Micro-Precision} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} (TP_i + FP_i)} \tag{9}$$

$$\text{Micro-Recall} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} (TP_i + FN_i)} \tag{10}$$

$$\text{Micro-F1} = \frac{2 \times \text{Micro-Precision} \times \text{Micro-Recall}}{\text{Micro-Precision} + \text{Micro-Recall}} \tag{11}$$

  In un contesto multi-classe, si può dimostrare che Micro-Precision = Micro-Recall = Micro-F1 = Accuracy. Questo perché ogni errore di classificazione (FN per una classe, FP per un'altra) viene contato una volta sia al numeratore che al denominatore delle frazioni aggregate globali in modo tale che le tre metriche coincidano. L'approccio micro dà più peso alle classi più popolose. È utile quando si vuole misurare le prestazioni complessive sul numero totale di istanze.

  Si può anche definire un Micro-BEP: [source: 28]

$$\mu BEP = \frac{\mu\text{Precision} + \mu\text{Recall}}{2} \tag{12}$$

Esempio di Calcolo (dalla matrice precedente):

- **Macro-Precision:** [source: 30] $\text{MPrecision} = \frac{1}{3}(\text{Precision}_A + \text{Precision}_B + \text{Precision}_C) = \frac{1}{3}(\frac{38}{49} + \frac{43}{55} + \frac{44}{46}) \approx \frac{1}{3}(0.775 + 0.782 + 0.957) \approx 0.838$ (Similmente per Macro-Recall e Macro-F1)

- **Micro-Average:** [source: 31, 58] $TP_{tot} = 38 + 43 + 44 = 125$ $FP_{tot} = (5 + 6) + (12 + 0) + (2 + 0) = 11 + 12 + 2 = 25$ $FN_{tot} = (12 + 0) + (5 + 2) + (6 + 0) = 12 + 7 + 6 = 25$ Numero totale di istanze $= 125 + 25 = 150$. $\mu\text{Precision} = \frac{\sum TP_i}{\sum (TP_i + FP_i)} = \frac{125}{125+25} = \frac{125}{150} \approx 0.833$ $\mu\text{Recall} = \frac{\sum TP_i}{\sum (TP_i + FN_i)} = \frac{125}{125+25} = \frac{125}{150} \approx 0.833$ $\mu F1 = \frac{2 \times 0.833 \times 0.833}{0.833 + 0.833} = 0.833$ Come previsto, $\mu\text{Precision} = \mu\text{Recall} = \mu F1 = \text{Accuracy}$.

4 Metodologie di Valutazione e Tuning

Oltre alle metriche, è fondamentale adottare procedure corrette per ottenere stime affidabili delle prestazioni di un modello su dati nuovi e sconosciuti.

4.1 Importanza dei Dati di Test e Overfitting

Le prestazioni di un modello misurate sui **dati di addestramento (training set)** non sono un indicatore affidabile delle sue capacità di generalizzazione. [source: 33] Un modello potrebbe imparare a memoria i dati di training, inclusi rumore e peculiarità specifiche di quel set, un fenomeno chiamato **overfitting**. [source: 34] Tale modello avrebbe prestazioni eccellenti sul training set ma scarse su dati nuovi (test set).

Per ottenere una stima realistica, la valutazione deve essere eseguita su un **test set** separato, contenente dati che il modello non ha mai visto durante l'addestramento. [source: 33]

4.2 Suddivisione del Dataset (Splitting)

Il processo tipico prevede la suddivisione del dataset disponibile in almeno due parti: [source: 35]

1. **Training Set:** Utilizzato per addestrare il modello (es. 70-80
2. **Test Set:** Utilizzato per valutare il modello finale (es. 20-30

Il processo di apprendimento e valutazione segue questi passi: [source: 36, 37]

1. **Addestramento (Learning Phase):** L'algoritmo di ML apprende i pattern dai dati del training set.
2. **Test (Testing Phase):** Il modello addestrato viene usato per fare previsioni sul test set.
3. **Valutazione (Evaluation):** Le previsioni del modello sul test set vengono confrontate con i valori reali (oracle) per calcolare le metriche di performance.

4.3 Validazione Incrociata (Cross-Validation)

Quando i dati sono scarsi, una singola suddivisione training/test potrebbe non essere rappresentativa. Le prestazioni misurate potrebbero dipendere fortemente dalla specifica suddivisione scelta. [source: 38] La **Validazione Incrociata (Cross-Validation, CV)**, in particolare la **k-Fold Cross-Validation**, è una tecnica più robusta per la valutazione: [source: 39]

1. Il dataset originale viene suddiviso in k sottoinsiemi (fold) di dimensioni approssimativamente uguali.
2. Il processo viene ripetuto k volte (round).
3. In ogni round i (da 1 a k):
   - Il i-esimo fold viene usato come **test set**.
   - I restanti $k-1$ fold vengono usati come **training set**.
   - Si addestra il modello sul training set e si valuta sul test set, calcolando le metriche desiderate.
4. Le metriche ottenute nei k round vengono mediate per ottenere una stima finale più stabile delle prestazioni del modello.

Un valore comune per k è 5 o 10. Un caso limite è il *Leave-One-Out Cross-Validation (LOOCV)*, dove k è uguale al numero totale di istanze.

4.4 Tuning dei Parametri e Validation Set

Molti algoritmi di ML hanno **iperparametri** che non vengono appresi dai dati ma devono essere impostati prima dell'addestramento (es. il valore k in k-NN, il tasso di apprendimento nelle reti neurali, i parametri β e γ o ρ nell'algoritmo di Rocchio [source: 41]). La scelta ottimale degli iperparametri è cruciale per le prestazioni. Il processo di **tuning** consiste nel trovare la configurazione migliore: [source: 41]

1. Definire un insieme di possibili valori (griglia) per gli iperparametri da testare.
2. Per ogni configurazione, addestrare e valutare il modello.

Attenzione: Il tuning non deve essere fatto usando il test set finale! Altrimenti, si finirebbe per "ottimizzare" gli iperparametri per quel specifico test set, introducendo un bias ottimistico nella valutazione finale (data leakage).

Per il tuning si introduce un terzo set di dati, il **Validation Set (o Tuning Set)**: [source: 42]

- **Training Set:** Usato per addestrare modelli con diverse configurazioni di iperparametri.
- **Validation Set:** Usato per valutare ciascun modello addestrato e scegliere la configurazione di iperparametri che dà le migliori prestazioni su questo set.
- **Test Set:** Usato *una sola volta alla fine*, per valutare le prestazioni del modello finale scelto (quello con la configurazione migliore, ri-addestrato eventualmente su training + validation set).

Quando si usa la Cross-Validation, il tuning può essere integrato nel processo ("nested cross-validation"): il training set di ogni fold viene ulteriormente suddiviso in training e validation per il tuning interno a quel fold.

Processo Completo: [source: 42] Dati -> Suddivisione in Training/Validation/Test -> Ciclo di Tuning (Addestra su Training, Valuta su Validation) -> Scelta Miglior Configurazione -> Addestramento Modello Finale (su Training o Training+Validation) -> Valutazione Finale (su Test Set).

5 Caso di Studio: Classificazione Reuters con Rocchio Parametrizzato

Per illustrare l'applicazione pratica delle metriche e delle metodologie di valutazione, consideriamo un esempio tratto dalla classificazione di testi, utilizzando il dataset Reuters-21578 e confrontando diversi classificatori, tra cui una versione parametrizzata dell'algoritmo di Rocchio. [source: 43, 46]

5.1 Algoritmo di Rocchio Parametrizzato (PRC)

L'algoritmo di Rocchio classico per la classificazione calcola un prototipo (centroide) per ogni classe C^i come combinazione lineare dei vettori dei documenti appartenenti alla classe (T_i) e quelli non appartenenti ($\overline{T}_i$). La formula tradizionale include parametri β e γ per pesare l'importanza dei documenti positivi e negativi.

Il lavoro di Basili et al. (IJCAI 2001/IJAIT 2002) propone una versione parametrizzata (PRC) che semplifica la formula e rende il parametro dipendente dalla classe C^i: [source: 44]

$$C_f^i = \max \left\{ 0, \frac{1}{|T_i|} \sum_{d \in T_i} d_f - \frac{\rho_i}{|\overline{T}_i|} \sum_{d \in \overline{T}_i} d_f \right\} \tag{13}$$

dove:

- C_f^i è il peso della feature (termine) f nel prototipo della classe i.
- d_f è il peso della feature f nel documento d (es. TF-IDF).
- $|T_i|$ e $|\overline{T}_i|$ sono le cardinalità degli insiemi di documenti positivi e negativi per la classe i.
- ρ_i è il parametro specifico per la classe i, che bilancia l'influenza dei documenti negativi.

Un aspetto interessante è che aumentando ρ_i, il secondo termine (sottrattivo) cresce, portando più pesi C_f^i a diventare zero. Questo agisce come una forma di **feature selection**, eliminando feature poco discriminative per quella classe. [source: 44] L'obiettivo è trovare il valore ottimale di ρ_i per ciascuna classe.

5.2 Setup Sperimentale e Tuning

Gli esperimenti sono stati condotti sul dataset Reuters-21578 (versione Apté split), con 90 classi, 9603 documenti di training e 3299 di test. [source: 46]

La procedura di stima del parametro ρ_i ottimale per ogni classe C^i è la seguente: [source: 64]

1. Creare un **validation set** (circa 30
2. Per un range di valori di ρ (es. da 0 a 30):
    - Addestrare il classificatore PRC sul restante 70
    - Misurare il Break-Even Point (BEP) sul validation set.
3. Selezionare il valore ρ_i^* che massimizza il BEP per la classe i sul validation set.
4. Ri-addestrare il classificatore PRC sull'intero training set originale usando i ρ_i^* trovati per ogni classe.
5. Testare il modello finale parametrizzato sul test set tenuto da parte.

Per risultati più robusti, questa procedura viene ripetuta usando una cross-validation (es. 20 fold) per la creazione dei validation set e la stima dei parametri ρ. [source: 64]

Le figure [Image 4] - [Image 8] mostrano l'impatto del parametro ρ sul BEP per diverse categorie del dataset Reuters, evidenziando come il valore ottimale di ρ vari significativamente tra classi più o meno popolate o complesse. [source: 59, 60, 61, 62, 63]

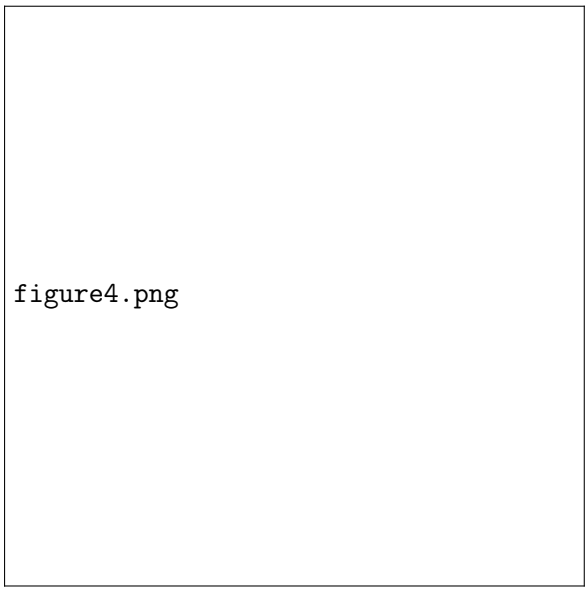

Figura 4: Impatto di ρ su BEP (Acquisition).

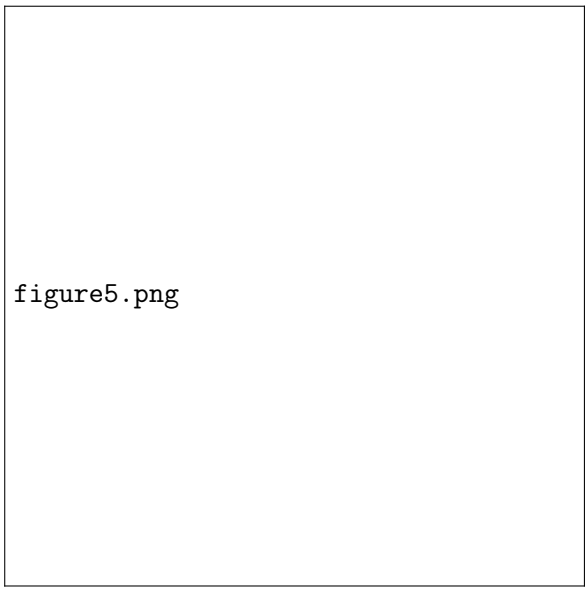

Figura 5: Impatto di ρ su BEP (Trade).

5.3 Risultati Comparativi

I risultati ottenuti con PRC sono stati confrontati con l'algoritmo di Rocchio standard (con valori fissi di β, γ) e con altri classificatori noti come le Support Vector Machines (SVM). [source: 65, 66, 67]

Risultati su Split Fisso Reuters (Micro-F1 $\approx \mu$BEP):

Feature Set	Rocchio Std ($\gamma = 1/4\beta$ o $\gamma = \beta$)	PRC	SVM
Tokens ($\sim$30k)	72.71% - 78.79%	**82.83%**	85.34%
Letteratura (stems)	75% - 79.9%	-	84.2%

PRC migliora significativamente rispetto a Rocchio standard e si avvicina alle prestazioni di SVM su questo task. [source: 67]

Ulteriori risultati da Cross-Validation (20 campioni) confermano la robustezza di PRC e permettono un confronto più dettagliato per classe e in termini di deviazione standard. [source: 78, 79] (La tabella completa è omessa per brevità ma presente nelle slide originali). L'analisi cross-validation mostra che PRC ottiene prestazioni significativamente migliori di Rocchio standard e competitive con SVM su molte categorie. [source: 79]

Questo caso di studio evidenzia l'importanza del tuning degli iperparametri (come ρ_i) e l'efficacia delle metodologie di valutazione rigorose (uso di validation set, cross-validation) per confrontare e migliorare i modelli di ML.

6 Diagnostica degli Errori: Bias e Varianza

Quando un sistema di ML ha prestazioni scarse, è fondamentale diagnosticare la causa del problema per poter applicare le giuste strategie correttive. [source: 82] Due problemi opposti ma comuni sono l'alto Bias e l'alta Varianza.

6.1 Bias vs. Varianza

- **Bias (Sottostima / Underfitting):** Si verifica quando il modello è **troppo semplice** per catturare la complessità sottostante dei dati. [source: 84] Il modello non riesce ad adattarsi bene nemmeno ai dati di addestramento.
  - **Sintomi:** Errore alto sia sul training set che sul test set. [source: 84]
  - **Causa:** Modello non sufficientemente espressivo (es. cercare di fittare dati non lineari con un modello lineare [source: 83]).
- **Varianza (Sovrastima / Overfitting):** Si verifica quando il modello è **troppo complesso** e si adatta eccessivamente ai dati di addestramento, imparando anche il rumore e le specificità del training set. [source: 84]
  - **Sintomi:** Errore molto basso sul training set, ma errore significativamente più alto sul test set. Grande divario tra errore di training e di test. [source: 84]
  - **Causa:** Modello troppo flessibile (es. un albero di decisione molto profondo, un polinomio di grado elevato [source: 83, 86]). Il modello non generalizza bene a dati nuovi.

L'obiettivo è trovare un modello con un buon compromesso tra Bias e Varianza.

6.2 Diagnosi tramite Curve di Apprendimento

Le **curve di apprendimento (learning curves)** sono grafici che mostrano l'andamento dell'errore (o di un'altra metrica) sul training set e sul validation/test set al variare della dimensione del training set (m). Sono utili per diagnosticare problemi di Bias e Varianza.

- **Diagnosi di Alto Bias (Underfitting):** [source: 85]
  - L'errore di training parte basso e cresce leggermente all'aumentare di m.
  - L'errore di test parte alto e decresce fino a raggiungere un plateau vicino all'errore di training.
  - **Caratteristica chiave:** Entrambi gli errori convergono a un valore **alto**. Il gap tra i due errori è piccolo.
  - **Interpretazione:** Il modello è troppo semplice. Anche con molti dati, non riesce a fittare bene. Aggiungere altri dati *non aiuta* significativamente.
- **Diagnosi di Alta Varianza (Overfitting):** [source: 86]
  - L'errore di training parte molto basso e rimane basso (o cresce molto lentamente) all'aumentare di m.
  - L'errore di test parte molto alto e decresce all'aumentare di m, ma rimane significativamente più alto dell'errore di training.
  - **Caratteristica chiave:** C'è un **grande gap** tra l'errore di training (basso) e l'errore di test (alto).
  - **Interpretazione:** Il modello è troppo complesso e si adatta troppo al training set. Il punto di saturazione non è ancora stato raggiunto. Aggiungere altri dati *potrebbe aiutare* a ridurre il gap e l'errore di test. [source: 87]

6.3 Tabella Riassuntiva: Bias vs. Varianza

Interpretando "PI" come Problema di Inadeguatezza (Bias) e "VI" come Problema di Varianza (Variance), ecco una tabella comparativa:

Bias (Sottostima / Underfitting)	**Varianza (Sovrastima / Overfitting)**
Il modello è troppo semplice.	Il modello è troppo complesso.
Non cattura la struttura dei dati.	Si adatta troppo al rumore/specificità del training set.
Errore alto su training set.	Errore basso su training set.
Errore alto su test set.	Errore alto su test set.
Piccolo gap tra errore training e test.	Grande gap tra errore training e test.
Diagnosi tramite Curva di Apprendimento	
Errore training e test convergono a un valore alto.	Grande divario tra errore training (basso) e test (alto).
Aggiungere dati non aiuta molto.	Aggiungere dati può aiutare (riduce il gap).
Possibili Soluzioni [source: 88, 89, 90]	
- Usare un modello più complesso (es. più parametri, polinomio di grado maggiore, rete neurale più profonda).	- Usare più dati di addestramento (se possibile, o usare data augmentation).
- Aggiungere nuove feature informative (ingegneria delle feature).	- Usare un modello più semplice (es. meno parametri, regolarizzazione).
- Ridurre la regolarizzazione.	- Rimuovere feature irrilevanti o rumorose (feature selection).
- Cambiare algoritmo di apprendimento.	- Aumentare la regolarizzazione.

6.4 Strategie di Miglioramento

Una volta diagnosticato il problema principale (Bias o Varianza), si possono applicare strategie mirate: [source: 88]

Se il problema è l'Alto Bias (Underfitting): [source: 89]

- Provare un modello più potente/complesso (es. aggiungere layer/neuroni in una rete neurale, usare kernel polinomiali in SVM, ridurre k in k-NN).
- Effettuare feature engineering per creare feature più significative.
- Ridurre o eliminare la regolarizzazione.

Se il problema è l'Alta Varianza (Overfitting): [source: 90]

- Raccogliere più dati di addestramento.
- Utilizzare tecniche di Data Augmentation per creare artificialmente nuovi dati.
- Provare un modello meno complesso (es. ridurre grado polinomio, usare meno feature, aumentare k in k-NN).
- Introdurre o aumentare la regolarizzazione (L1, L2, Dropout, etc.).
- Effettuare feature selection per rimuovere feature poco utili o rumorose.

È un processo iterativo: si applica una modifica, si rivaluta il modello (usando le curve di apprendimento e le metriche sul validation set) e si decide il passo successivo.

7 Conclusioni

La valutazione delle prestazioni è una fase imprescindibile nel ciclo di vita di un sistema di Machine Learning. [source: 91] Comprendere e utilizzare correttamente metriche come Accuracy, Precision, Recall, F1-score, insieme alle loro varianti aggregate (Macro/Micro average) e a strumenti come la matrice di confusione e le curve Precision-Recall, permette di misurare oggettivamente l'efficacia di un modello.

Tuttavia, le metriche da sole non bastano. È cruciale adottare metodologie di valutazione rigorose, come la separazione netta tra training, validation e test set, e l'uso della cross-validation per ottenere stime affidabili e ridurre la dipendenza da specifiche suddivisioni dei dati. [source: 92]

Infine, la diagnostica degli errori, in particolare la distinzione tra problemi di Bias (Underfitting) e Varianza (Overfitting) tramite l'analisi delle curve di apprendimento, fornisce indicazioni preziose su come intervenire per migliorare le prestazioni del sistema, guidando la scelta tra l'adozione di modelli più o meno complessi, la raccolta di più dati o l'ingegnerizzazione delle feature. [source: 92] La scelta della metrica e della strategia di miglioramento più adatta dipende fortemente dal contesto applicativo e dagli obiettivi specifici del sistema di ML.