TOR VERGATA

UNIVERSITÀ DEGLI STUDI DI ROMA

Dipartimento di Ingegneria Civile e Ingegneria Informatica

Metodi di Ottimizzazione Per Big Data

Relazione progetto

Partecipanti:

Lorenzo Grande 0350212

A.A 2024-2025

Contents

Chapter 1

Introduzione

Il progetto di Metodi di Ottimizzazione per Big Data prevede l'implementazione di un modello di machine learning. A tal proposito, nel seguente progetto, è stata implementata una rete neurale per un problema di classificazione binaria.
Il linguaggio scelto è Python e al fine di eseguire il modello è necessario scaricare le seguenti librerie:

- numpy
- matplotlib
- pandas
- scikit-learn
- imbalanced-learn

Chapter 2

Architettura

2.1 Rete Neurale

Una rete neurale è un modello computazionale ispirato al funzionamento del cervello umano, composto da unità elementari chiamate **neuroni**. Questi neuroni ricevono informazioni in ingresso, le elaborano attraverso funzioni matematiche e le trasmettono ad altri neuroni.

Una rete neurale è generalmente suddivisa in tre componenti principali:

- **Input layer**: Lo strato d'ingresso che riceve i dati del problema;
- **Hidden layers**: Strati intermedi che elaborano e trasformano le informazioni attraverso i neuroni e le connessioni, consentendo alla rete di apprendere relazioni complesse nei dati;
- **Output layer**: Lo strato d'uscita che produce l'output della rete

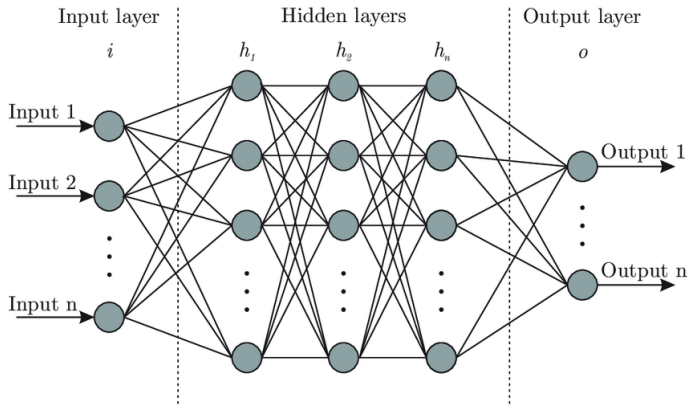


Figure 2.1: Struttura generica di una rete neurale.

Nel progetto descritto, è stata implementata una rete neurale specifica per la classificazione binaria. La struttura della rete è la seguente:

- **Input layer**: N neuroni, pari al numero di feature del dataset;
- **Primo hidden layer**: 16 neuroni;
- **Secondo hidden layer**: 32 neuroni;
- **Output layer**: 1 neurone

Tale configurazione è stata scelta in quanto rappresenta un buon compromesso fra capacità di apprendere e complessità computazionale.

2.2 Funzioni di attivazione

Le funzioni di attivazione svolgono un ruolo fondamentale nelle reti neurali, in quanto determinano come le informazioni vengono elaborate all'interno della rete, aggiungendo non linearità al modello e consentendo così di apprendere relazioni complesse nei dati. Nel seguente progetto è possibile scegliere fra due funzioni di attivazione per gli strati nascosti: **ReLU** e **tanh**. Per l'ultimo strato della rete, invece, viene utilizzata la funzione di attivazione **sigmoid**. Questa scelta è motivata dalla capacità della funzione sigmoid di mappare i valori in un intervallo (0,1), rendendola ideale nel contesto di classificazione binaria, in cui l'output può essere interpretato come probabilità di appartenenza alla classe positiva.

2.3 Inizializzazione pesi e addestramento della rete neurale

2.3.1 Inizializzazione pesi

L'obiettivo di una rete neurale è quello di apprendere a mappare gli input agli output desiderati attraverso un processo di ottimizzazione. Per raggiungere questo scopo, è necessario addestrare la rete, ossia trovare i parametri (pesi) che minimizzano una funzione di errore opportunamente definita. Prima di tutto, è fondamentale inizializzare i pesi, in quanto questa scelta influisce direttamente sulla capacità del modello di convergere verso una soluzione ottimale. Inizializzare i pesi con valori casuali piccoli è essenziale per rompere la simmetria tra i nodi della rete, evitando che tutti gli strati apprendano gli stessi valori e prevenendo una convergenza troppo lenta o inefficace [1].

Nel contesto di questo progetto, si è scelto di utilizzare due tipi di inizializzazione:

- **Inizializzazione di He e Xavier**: Entrambe le tecniche sono state sviluppate per risolvere problemi legati alla convergenza nelle reti neurali, rispettivamente per reti con funzioni di attivazione ReLU e tanh [2].

2.4 Addestramento

2.4.1 Backpropagation

L'obiettivo della fase di training è quello di minimizzare la funzione di costo rispetto ai parametri della rete neurale. Per raggiungere tale obiettivo, è stato utilizzato l'algoritmo di **backpropagation**:

- Innanzitutto, è stata realizzata una fase di **forward propagation** per calcolare l'uscita della rete e l'errore commesso rispetto alle etichette reali;
- Successivamente, tramite la fase di **backward propagation**, è stato possibile calcolare i gradienti della funzione di costo rispetto ai pesi e ai bias della rete.

2.4.2 Funzione di Costo

Come funzione di costo, è stata utilizzata la **binary cross-entropy**:

$$L = -\frac{1}{|B|} \sum_{i \in B} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Dove:

- $|B|$ è la dimensione del mini-batch considerato;
- y_i rappresenta l'etichetta vera del campione i-esimo;
- $\hat{y}_i$ rappresenta l'output della rete, ossia la probabilità di appartenenza alla classe positiva.

2.4.3 Aggiornamento dei Parametri

Per l'aggiornamento dei parametri, è stato utilizzato il metodo di ottimizzazione **Stochastic Gradient Descent con momentum**. I pesi vengono aggiornati come segue:

$$w_{k+1} = w_k - \alpha_k \cdot \tilde{\nabla} R(w_k) + \beta(w_k - w_{k-1})$$

Dove:

- α_k è il learning rate, calcolato tramite la tecnica del diminishing stepsize. L'uso efficace di tale tecnica consente al modello di apprendere rapidamente all'inizio (quando il valore di α_k è pari a 0.01), per poi procedere in modo più incrementale, evitando di superare la convergenza;
- β rappresenta il parametro che determina quanto tenere conto delle iterazioni precedenti per aggiornare i parametri. Il valore di β è stato fissato a 0.9.

2.4.4 Mini-batch

Nel progetto, il set di campioni di training è stato suddiviso in mini-batch di dimensione fissa pari a 64. Questa tecnica risulta particolarmente utile in caso di dataset di grandi dimensioni, poiché gran parte delle informazioni provenienti dai dati può essere ridondante. L'uso di tutti i campioni in un singolo aggiornamento potrebbe risultare inefficiente anche dal punto di vista del calcolo del gradiente.

2.5 Regolarizzazione e Strategia di Cross-Validation

Regolarizzazione

Per evitare il fenomeno di **overfitting**, in cui il modello fornisce ottimi risultati sui campioni del training set ma pessimi sui nuovi dati, possiamo utilizzare tecniche di regolarizzazione che penalizzano la complessità del modello. Nel nostro progetto è possibile scegliere tra due tipi di regolarizzazione:

- **Regolarizzazione L2**: consente di avere parametri piccoli in valore assoluto, ma diversi da zero;

- **Regolarizzazione L1**: promuove la sparsità, ossia molti valori pari a zero. Questa caratteristica permette anche l'implementazione di metodi di **feature selection** per selezionare le caratteristiche più importanti.

Cross-Validation

La strategia di **cross-validation**[3] prevede:

- La suddivisione del dataset in training set, validation set e test set;
- Il fissaggio di una griglia di valori per il parametro di regolarizzazione λ, in particolare:
  - Per la regolarizzazione L2: $\lambda \in$ [1e-2, 1e-1, 0.1, 0.15, 0.3];
  - Per la regolarizzazione L1: $\lambda \in$ [1e-4, 1e-3, 0.01, 0.1, 0.3].
- Per ogni valore di λ, addestriamo il modello utilizzando i dati del training set e stimiamo l'errore sul validation set;
- Scegliamo il valore di λ che minimizza l'errore sul validation set;
- Utilizziamo il test set per valutare le prestazioni finali del modello.

Chapter 3

Dataset

3.1 Analisi dei Dataset

Per valutare le performance del modello, sono stati utilizzati i seguenti dataset:

- **WineQuality Dataset:** [4] Questo dataset include alcune caratteristiche fisico-chimiche di vini rossi prodotti nel nord del Portogallo. In particolare, permette di determinare se un vino può essere classificato come "di qualità" o meno sulla base di queste misurazioni.

  Le principali caratteristiche del dataset sono:

  - **Numero di campioni**: 1599
  - **Numero di features**: 11
  - **Numero di campioni appartenenti alla classe positiva**: 744

- **Mushroom Dataset:** [5] Questo dataset raccoglie informazioni su alcune caratteristiche chiave dei funghi, come il diametro e la forma del cappello, la stagione di raccolta e altre proprietà distintive. L'obiettivo principale è determinare se un fungo è velenoso o commestibile.

  Le principali caratteristiche del dataset sono:

  - **Numero di campioni**: 54,035
  - **Numero di features**: 8
  - **Numero di campioni appartenenti alla classe positiva**: 29,675

- **Airline Satisfaction Dataset:** [6] Questo dataset raccoglie i risultati di un questionario compilato da clienti di diverse compagnie aeree. L'obiettivo è determinare, in base a vari fattori come il tipo di viaggio, la durata del volo, la qualità dei sedili e molte altre caratteristiche, se i clienti sono stati soddisfatti o meno del loro volo.

  Le principali caratteristiche del dataset sono:

  - **Numero di campioni**: 129,880
  - **Numero di features**: 25
  - **Numero di campioni appartenenti alla classe positiva**: 56,428

  Dato l'elevato numero di features, è fondamentale valutare se tutte contribuiscono efficacemente alla classificazione della soddisfazione del cliente o se alcune possono essere eliminate senza perdita di accuratezza.

3.2 Preprocessamento dei dataset

Per preparare i dataset all'addestramento del modello di rete neurale, sono state effettuate una serie di operazioni di preprocessamento.

3.2.1 Trasformazione del dataset

Nella prima fase del preprocessamento, i dati sono stati trasformati in una forma coerente con le esigenze della nostra implementazione e pronti per essere utilizzati nella fase di classificazione. Le operazioni di preprocessamento eseguite sui diversi dataset in questa fase sono le seguenti:

- **WineQuality Dataset:** Il dataset inizialmente includeva un punteggio di qualità che andava da 0 a 10, dove 0 rappresentava una qualità minima e 10 una qualità massima. Per adattarlo alla classificazione binaria, è stata effettuata una trasformazione: i vini con un punteggio maggiore di 5 sono stati classificati come di "alta qualità" (etichetta 1), mentre i vini con punteggio inferiore o uguale a 5 sono stati classificati come "bassa qualità" (etichetta 0).
- **Mushroom Dataset:** Il dataset presentava già una struttura adatta per l'elaborazione.
- **Airline Satisfaction Dataset:** In questo caso, il dataset conteneva numerosi valori in formato stringa, soprattutto nelle caratteristiche legate alla soddisfazione del cliente, come il tipo di viaggio o la qualità dei servizi. È stato quindi necessario mappare questi valori in

classi numeriche: per esempio, la variabile *Satisfaction* è stata trasformata in una variabile binaria, dove "satisfied" è stato mappato a 1 e "neutral or dissatisfied" a 0.

3.2.2 Preprocessing sui dati

Successivamente, è stata effettuata una fase di pre-processing sui dati per migliorarne la qualità e prepararli per l'addestramento del modello. In questa fase sono stati applicati i seguenti interventi:

- **Rimozione dei valori target NaN:** Sono stati eliminati tutti i campioni che presentavano un valore NaN nell'etichetta target;
- **Separazione delle features e delle etichette:** I dati sono stati separati in due set distinti: uno contenente le *features* e l'altro contenente il valori *target* di ciasucn campione;
- **Sostituzione dei valori mancanti (NaN) con la mediana della colonna:** Nel caso in cui fossero presenti valori mancanti (NaN) tra le features, questi sono stati sostituiti con la mediana della rispettiva colonna;
- **Oversampling delle classi minoritarie in caso di sbilanciamento:** Questa tecnica è stata applicata per limitare il più possibile il problema dello sbilanciamento delle classi. In caso di sbilanciamento, il modello potrebbe prediligere la classe maggioritaria, commettendo errori nelle previsioni. Per mitigare il problema, è stata utilizzata la tecnica di oversampling, che aumenta artificialmente i campioni della classe minoritaria. In particolare, è stato utilizzato SMOTE, una tecnica che genera campioni sintetici a partire da un campione casuale della classe minoritaria [7].
- **Normalizzazione delle features:** Le variabili continue sono state normalizzate per garantire che tutte le features avessero lo stesso ordine di grandezza e fossero quindi comparabili tra loro.
- **Divisione del dataset in set di addestramento, validazione e test:** Il dataset è stato suddiviso in tre sottoinsiemi: uno per l'addestramento del modello (*training set*), uno per la validazione durante l'addestramento (*validation set*) e uno per il test finale del modello (*test set*). La divisione è stata effettuata con le seguenti proporzioni: il 20% dei dati è stato riservato per il validation set, mentre il 10% è stato destinato al test set.

Chapter 4

Risultati

4.1 Valutazione del modello

Per valutare le prestazioni del modello sono state considerate le seguenti metriche:

- **Accuratezza**: Misura la percentuale di previsioni corrette rispetto al totale dei dati analizzati.
- **Recall**: Rappresenta la capacità del modello di identificare correttamente tutti i dati appartenenti alla classe positiva:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

  Dove:

  - **TP** (True Positive): rappresenta i dati appartenenti alla classe positiva etichettati correttamente;
  - **FN** (False Negative): rappresenta i dati appartenenti alla classe positiva ma erroneamente etichettati come appartenenti alla classe negativa.

- **Precision**: Indica la proporzione di previsioni positive che sono effettivamente corrette:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

  Dove:

  - **FP** (False Positive): rappresenta i dati etichettati erroneamente come appartenenti alla classe positiva.

- **F1-Score**: Questa metrica combina *Precision* e *Recall* in un unico valore, calcolato come media armonica, enfatizzando maggiormente il valore più basso tra i due. L'F1-Score è particolarmente utile in scenari con classi sbilanciate:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

4.2 Risultati

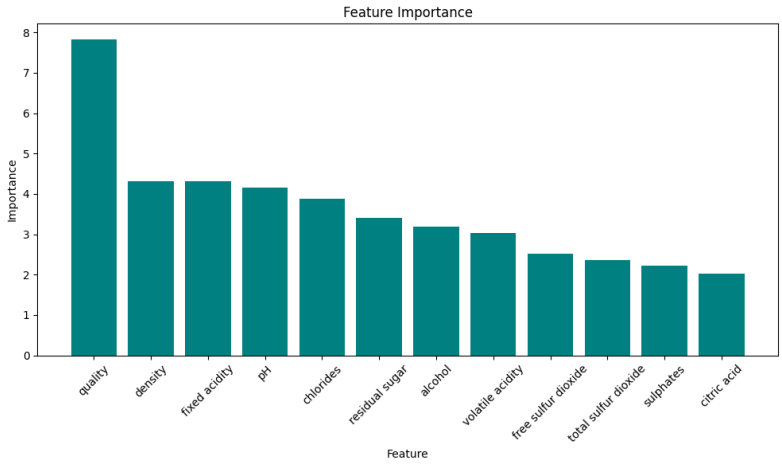
4.2.1 WineQuality Dataset

Errore di Preprocessamento nel WineQuality Dataset

Come descritto nella subsection 3.2.1, il dataset originale includeva un punteggio di qualità compreso tra 0 e 10, dove 0 rappresentava una qualità minima e 10 una qualità massima. Per adattare il dataset a una classificazione binaria, è stato effettuato un preprocessamento che ha trasformato il punteggio in un flag binario: i vini con punteggio superiore a 5 sono stati classificati come "alta qualità" (etichetta 1), mentre quelli con punteggio pari o inferiore a 5 come "bassa qualità" (etichetta 0).

Tuttavia, durante questa fase è stato commesso un errore: il punteggio originale di qualità è stato erroneamente mantenuto nel dataset insieme al flag binario. Questo ha introdotto un fenomeno noto come *label leakage*, poiché il modello aveva accesso diretto a una feature che rappresentava implicitamente la variabile target. Di conseguenza, il modello ha raggiunto una precisione del 100%, un risultato non realistico che non rifletteva le sue reali capacità predittive.

L'anomalia è stata individuata grazie all'osservazione di un'accuratezza così elevata, un valore inconsueto considerando anche le dimensioni limitate del dataset. Per confermare il sospetto di *label leakage*, è stata applicata una selezione delle feature tramite regolarizzazione L1, visualizzando le feature più rilevanti per il modello.



Metrica	Valore
Accuracy	100.0%
Precision	100.0%
Recall	100.0%
F1 Score	1.0

Figure 4.1: Risultati ottenuti con il test set.

Dalla figura, si osserva chiaramente come la feature `quality` (che rappresenta il punteggio originale) sia risultata la più significativa.

WineQualityCorrect Dataset

ReLU

Successivamente, è stata eseguita la cross-validation sul dataset corretto, ottenendo che il risultato migliore si ottiene utilizzando la regolarizzazione L2 con $\lambda = 0.1$.

Tempo impiegato per il training: 3.507 secondi.

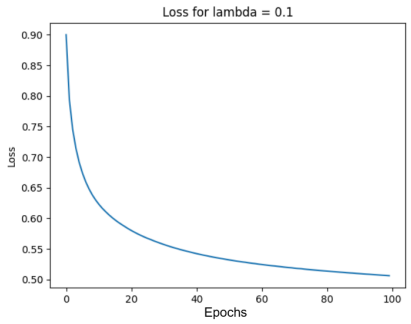


Figure 4.2: Loss durante l'addestramento.

Metrica	Valore
Accuracy	76.875%
Precision	79.26%
Recall	78.40%
F1 Score	0.789

Figure 4.3: Risultati ottenuti con il test set.

Tanh

Eseguendo la cross-validation, si osserva che il risultato migliore si ottiene utilizzando la regolarizzazione L2 con $\lambda = 0.01$.

Tempo impiegato per il training: 3.506 secondi.

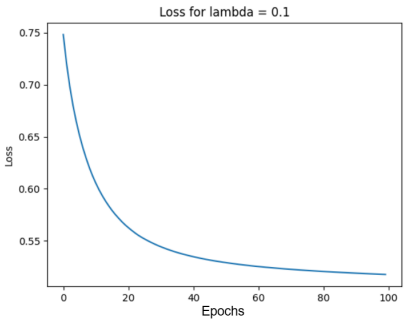


Figure 4.4: Loss durante l'addestramento.

Metrica	Valore
Accuracy	76.25%
Precision	78.40%
Recall	78.40%
F1 Score	0.784

Figure 4.5: Risultati ottenuti con il test set.

4.2.2 Mushroom Dataset

ReLU

Eseguendo la cross-validation, si osserva che il risultato migliore si ottiene utilizzando la regolarizzazione L1 con $\lambda = 0.0001$.

Tempo impiegato per il training: 127.82 secondi.

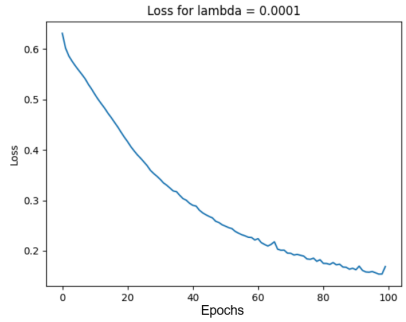


Figure 4.6: Loss durante l'addestramento.

Metrica	Valore
Accuracy	94.67%
Precision	93.92%
Recall	96.52%
F1 Score	0.95

Figure 4.7: Risultati ottenuti con il test set.

Tanh

Eseguendo la cross-validation, si osserva che il risultato migliore si ottiene utilizzando la regolarizzazione L1 con $\lambda = 0.001$.

Tempo impiegato per il training: 121.21 secondi.

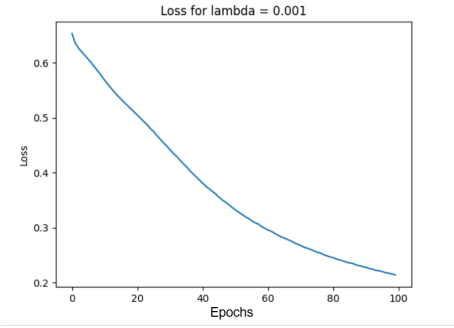


Figure 4.8: Loss durante l'addestramento.

Metrica	Valore
Accuracy	91.49%
Precision	92.75%
Recall	91.63%
F1 Score	0.92

Figure 4.9: Risultati ottenuti con il test set.

4.2.3 Airline Satisfaction Dataset

ReLU

Eseguendo la cross-validation, si osserva che il risultato migliore si ottiene utilizzando la regolarizzazione L2 con $\lambda = 0.01$.

Tempo impiegato per il training: 265.65 secondi.

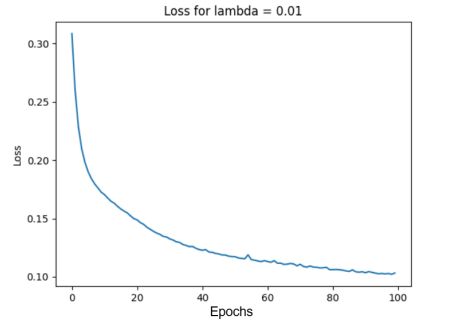


Figure 4.10: Loss durante l'addestramento.

Metrica	Valore
Accuracy	95.40%
Precision	96.85%
Recall	92.45%
F1 Score	0.95

Figure 4.11: Risultati ottenuti con il test set.

Tanh

Eseguendo la cross-validation, si osserva che il risultato migliore si ottiene utilizzando la regolarizzazione L2 con $\lambda = 0.01$.

Tempo impiegato per il training: 256.79 secondi.

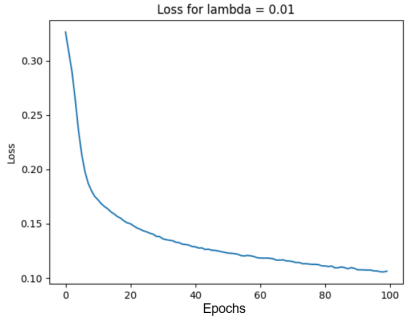


Figure 4.12: Loss durante l'addestramento.

Metrica	Valore
Accuracy	95.18%
Precision	95.85%
Recall	93.10%
F1 Score	0.94

Figure 4.13: Risultati ottenuti con il test set.

Airline Satisfaction Dataset: Features Selection

Il seguente dataset contiene una grande quantità di features, quindi vale la pena chiedersi se alcune di esse siano meno importanti di altre. Eseguendo una regolarizzazione L1 e applicando un meccanismo di feature selection otteniamo:

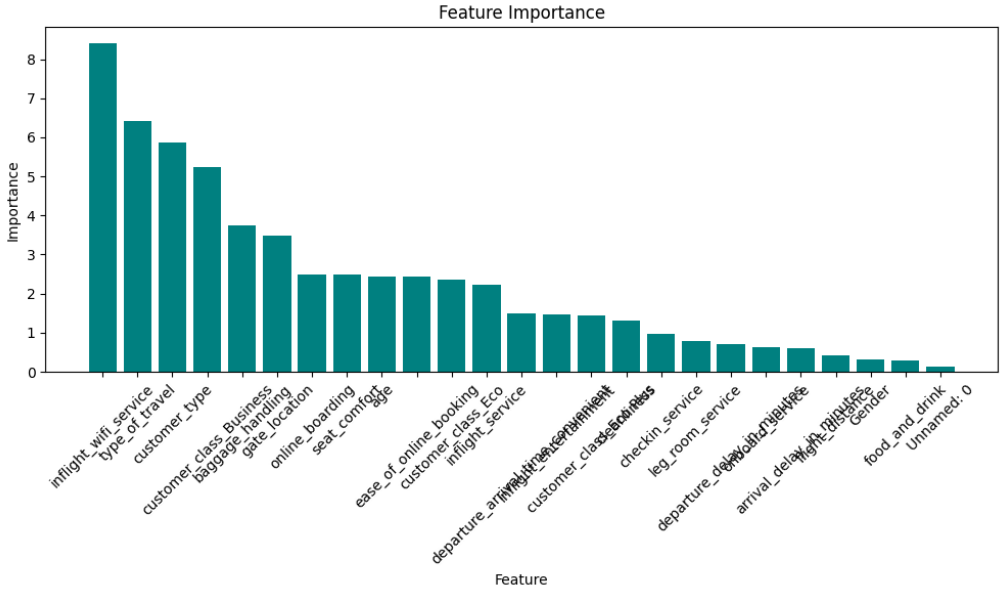


Figure 4.14

Eseguiamo quindi nuovamente il training sul dataset escludendo le tre features meno importanti:

Relu

Eseguendo la cross-validation, si osserva che il risultato migliore si ottiene utilizzando la regolariz-

zazione L2 con $\lambda = 0.01$.

Tempo impiegato per il training: 264.32 secondi.

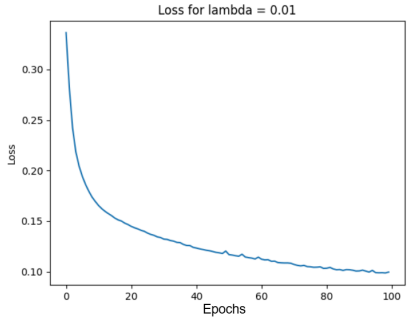


Figure 4.15: Loss durante l'addestramento.

Metrica	**Valore**
Accuracy	95.42%
Precision	95.36%
Recall	94.08%
F1 Score	0.95

Figure 4.16: Risultati ottenuti con il test set.

Tanh Eseguendo la cross-validation, si osserva che il risultato migliore si ottiene utilizzando la regolarizzazione L2 con $\lambda = 0.01$.

Tempo impiegato per il training: 274.35 secondi.

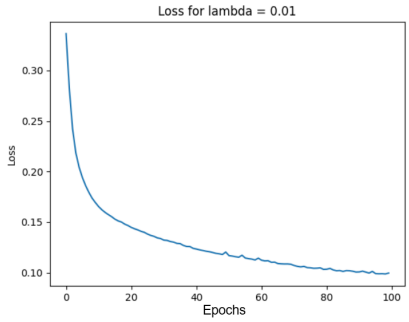


Figure 4.17: Loss durante l'addestramento.

Metrica	**Valore**
Accuracy	95.26%
Precision	95.75%
Recall	93.26%
F1 Score	0.94

Figure 4.18: Risultati ottenuti con il test set.

Osservazioni:

Notiamo come le prestazioni rimangano pressoché invariate nonostante l'eliminazione di 3 features, a conferma di quanto già osservato nella Figure 4.14.

4.3 Osservazioni Finali

Dai risultati ottenuti emergono alcuni aspetti fondamentali:

- **La dimensione del dataset influenza la capacità di apprendimento:** Questo è particolarmente evidente con il `WineQuality Dataset`, che presenta performance inferiori rispetto agli altri due dataset, probabilmente a causa del numero di campioni significativamente inferiore.
- **La cross-validation si dimostra una scelta migliore rispetto a un valore di λ fisso:** Nei nostri esperimenti, abbiamo osservato che la cross-validation ha portato a prestazioni migliori, esplorando diverse configurazioni di λ.
- **La funzione ReLU si conferma una scelta ottimale come funzione di attivazione:** Nei nostri esperimenti, l'uso della ReLU ha prodotto prestazioni superiori, seppur di poco, rispetto alla Tanh.

Bibliography

[1] M.V. Narkhede, P.P. Bartakke, and M.S. Sutaone. "A review on weight initialization strategies for neural networks". In: *Artificial Intelligence Review* 55.2 (2022), pp. 291–322. DOI: 10.1007/s10462-021-10033-z.

[2] Leonid Datta. "A Survey on Activation Functions and their relation with Xavier and He Normal Initialization". In: *arXiv* 2004.06632 (2020). Delft University of Technology. URL: https://arxiv.org/abs/2004.06632.

[3] Andrea Cristofari. *Metodi di Ottimizzazione per Big Data, Modelli e algoritmi di machine learning*. Slide del corso LM Ingegneria Informatica – a.a. 2022/2023. Dipartimento di Ingegneria Civile e Ingegneria Informatica, 2023.

[4] Paulo Cortez et al. *Wine Quality*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C56S3T. 2009.

[5] Prisha Sawhney. *Mushroom Dataset (Binary Classification)*. Dataset aggiornato 9 mesi fa. Classificazione binaria dei funghi in commestibili e velenosi. 2024. URL: https://www.kaggle.com/datasets/prishasawhney/mushroom-dataset.

[6] TJ Klein. *Airline Passenger Satisfaction*. Dataset aggiornato 5 anni fa. 2018. URL: https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction.

[7] Georgios Douzas, Fernando Bacao, and Felix Last. "Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE". In: *Information Sciences* 465 (2018), pp. 1–20. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2018.06.056. URL: https://www.sciencedirect.com/science/article/pii/S0020025518304997.