Relazione Progetto di Machine Learning
Classificazione di lettere dell'alfabeto ASL

Habasescu Alin Marian

January 21, 2025

Contents

1 Descrizione del Progetto

L'obiettivo di questo progetto è progettare un sistema di classificazione in grado di identificare lettere statiche dell'alfabeto americano dei segni (ASL) a partire da immagini di dimensione 28×28 pixel. Le lettere Y e Z sono escluse dall'analisi poichè richiedono movimento per essere rappresentate.

Sono stati considerati e confrontati diversi modelli di Machine Learning affrontati durante il corso, al fine di identificare il modello che ottiene i migliori risultati in termini di accuratezza e capacità di generalizzazione.

- **Naive Bayes**
- **Multi Layer Perceptron Classifier (MLPClassifier)**
- **Support Vector Machine (SVM)**
- **Decision Tree**

Metodologia Il progetto segue i seguenti passi principali:

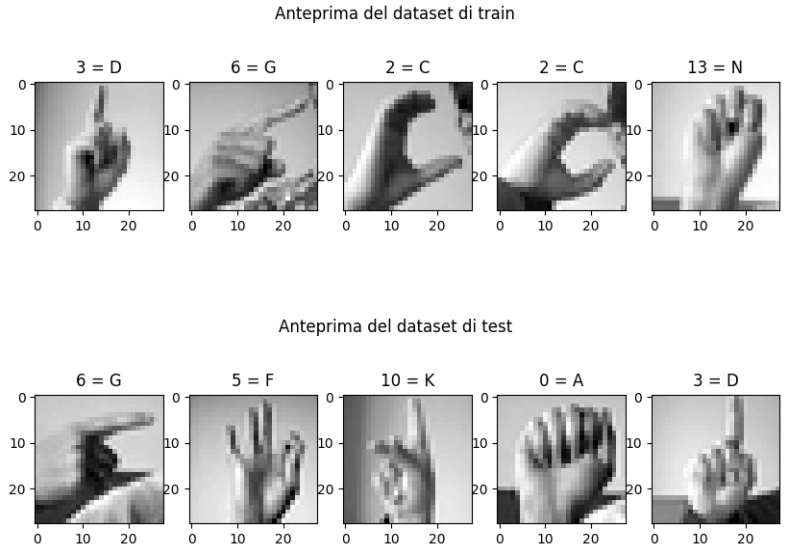
1. **Caricamento e visualizzazione dei dati**.
2. **Preprocessing dei dati**.
3. **Addestramento dei modelli** e **valutazione dei modelli**.
4. **Analisi dei Risultati ottenuti**.

2 I Dataset

Nel progetto si utilizzano due dataset distinti per l'addestramento e la valutazione dei modelli. I dataset vengono caricati dai file CSV forniti tramite la funzione **load_datasets**, che suddivide i dati in *features* (valori dei pixel delle immagini) e *target* (le lettere corrispondenti).

Il dataset di training è composto da 27455 campioni, mentre il dataset di test contiene 7172 campioni. Ogni campione rappresenta un'immagine di una lettera statica dell'alfabeto ASL, accompagnata dai valori di grigio relativi ai 784 pixel che costituiscono l'immagine.

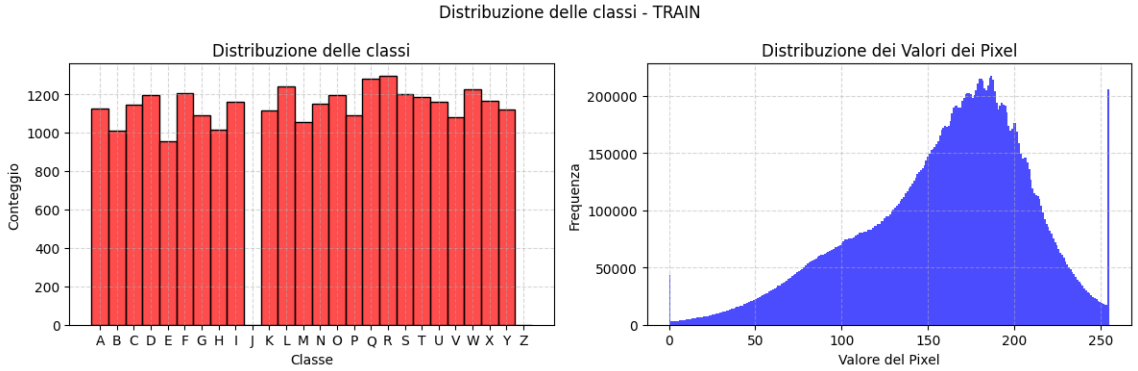
La funzione **visualize_dataset** consente di analizzare i dati caricati, mostrando un'anteprima grafica delle prime osservazioni sotto forma di immagini 28×28 pixel in scala di grigi e associandole alle lettere dell'alfabeto ASL (025, mappati su AZ).



3 Analisi Esplorativa dei Dati (EDA)

Dopo il caricamento dei dataset, viene eseguita una serie di operazioni per analizzare le caratteristiche dei dati. La funzione **perform_eda** viene utilizzata per eseguire l'analisi esplorativa dei dati.

- **Distribuzione delle classi**: Verifica di un bilanciamento uniforme tra le classi.
- **Distribuzione dei valori dei pixel**: Analisi della gamma di valori di pixel (0255) per determinare la necessità di normalizzazione.



Durante l'analisi esplorativa dei dati, non sono stati osservati evidenti squilibri nella distribuzione delle classi del dataset di training. Sono stati trovati due picchi [0, 255] nel grafico della distribuzione dei valori dei pixel, attribuibili ai valori di pixel molto bassi (0) e molto alti (255), rispettivamente.

4 Preprocessing dei dati

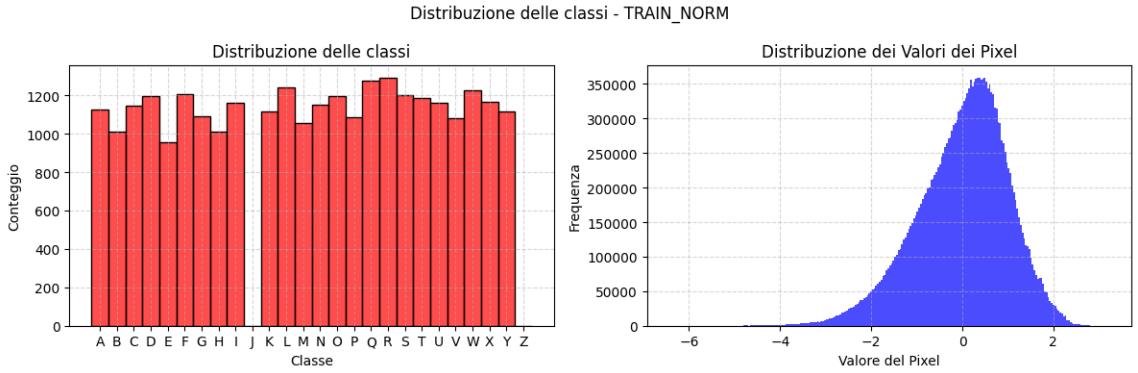
Il preprocessing è stato fatto in due fasi principali:

4.1 Standardizzazione

I dati sono stati standardizzati utilizzando `StandardScaler` di scikit-learn, che **normalizza i dati in modo che abbiano media 0 e varianza 1**.

Questo processo è fondamentale per:

- Normalizzare la scala dei pixel
- Migliorare la convergenza dei modelli
- Garantire che tutti i pixel contribuiscano euquivalentemente alla classificazione



4.2 Riduzione della dimensionalità

E' stata applicata l'analisi delle componenti principali (PCA) per:

- Riduzione della dimensionalità dei dati mantenendo il 95% della varianza
- Diminuire il rumore nei dati
- Migliorare l'efficienza computazionale in generale

5 Addestramento dei modelli e metriche di valutazione

Sono stati implementati e confrontati quattro modelli di Machine Learning ciascuno con caratteristiche specifiche:

5.1 Modelli Implementati

- **Naive Bayes**: Un modello di classificazione probabilistico che fa uso del Teorema di Bayes e dell'indipendenza tra le features.
- **MLPClassifier**: Una rete neurale a più strati con le seguenti caratteristiche
  - Architettura: tre layer nascosti (256, 128, 64 neuroni)
  - Funzione di attivazione ReLU
  - Ottimizzatore: Adam
  - Numero massimo di iterazioni: 5000
  - Regolarizzazione L2 con `alpha=0.0001`
- **Support Vector Machine (SVM)**: Un algoritmo di classificazione che trova un iperpiano ottimale che separa le classi. In questo progetto, l'algoritmo SVM utilizza il kernel Radial Basis Function (RBF) per proiettare i dati in uno spazio di alta dimensionalità e trovare l'iperpiano di separazione.
- **Decision Tree**: Un modello basato su una struttura ad albero, che suddivide i dati in base a regole decisionali sequenziali, fino a profondità massiama di 25.

Metriche di valutazione Per valutare le performance dei modelli sono state utilizzate le seguenti metriche di valutazione:

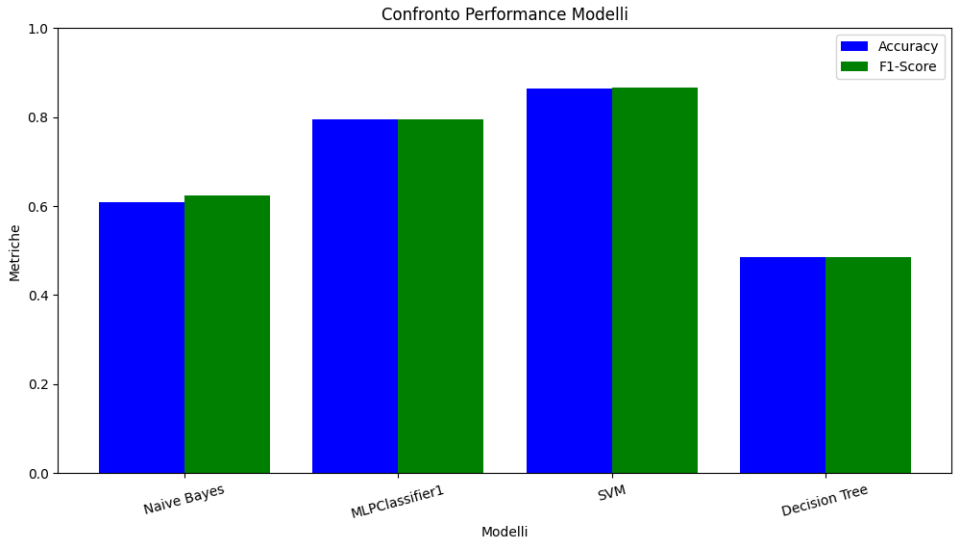
- **Accuracy**: Rappresenta la percentuale di campioni correttamente classificati.
- **F1_score**:
$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$
Media armonica tra precision[1] e recall[2]
- **Classification Report**: Precision, Recall e F1-Score per ogni classe.
- **Confusion Matrix**: Matrice di confusione che mostra il numero di campioni correttamente classificati per ogni classe.

I risultati sono stati confrontati tramite un grafico a barre che mostra le accuratezze dei modelli

[1] La percentuale di esempi correttamente classificati rispetto a tutti quelli predetti per una certa classe
[2] La percentuale di esempi correttamente classificati rispetto a tutti i veri esempi di una certa classe

6 Risultati ottenuti

L'analisi delle performance ha prodotto i seguenti risultati:



- **Support Vector Machine (SVM)** ha ottenuto l'accuratezza migliore (86%) tra i modelli analizzati, sebbene con un alto costo computazionale. Questo modello è stato particolarmente efficace nel trovare l'iperpiano ottimale grazie alla sua capacità di lavorare in spazi ad alta dimensione. Tuttavia, la sua performance è stata limitata dalla necessità di una grande quantità di risorse computazionali, rendendo la sua implementazione meno pratica in scenari a larga scala.
- **MLPClassifier** ha raggiunto un'accuratezza dell'80%, posizionandosi al secondo posto. Il modello ha avuto successo grazie alla sua capacità di catturare pattern complessi nelle immagini grazie alla struttura a più layer nascosti.
- **Naive Bayes** ha ottenuto un'accuratezza del (60%), principalmente a causa dell'assunzione di indipendenza tra le features. Questa ipotesi, non tiene conto che nelle immagini pixel vicini tendono ad avere valori di grigio simili.
- **Decision Tree** ha mostrato i risultati peggiori (43%). Il modello ha avuto difficoltà nel catturare pattern complessi, portando a performance insoddisfacenti rispetto ad altri modelli più sofisticati.