

# Relazione Progetto Machine Learning

Simone Sulis

Matricola: 60/79/00194

Simone Manuva

Matricola: 60/79/00184

Anno Accademico: 2024/2025

## 1 Introduzione

In questo progetto abbiamo analizzato il dataset `ObesityDataSet_raw_and_data_synthetic.csv` per la classificazione del livello di obesità di un individuo in base a parametri fisici e abitudini alimentari. L'obiettivo del progetto è confrontare diversi modelli di machine learning per ottenere la miglior accuratezza possibile e determinare la combinazione ottimale di tecniche di pre-processing e classificazione. Abbiamo implementato sia classificatori standard che personalizzati, ottimizzandoli tramite tuning degli iperparametri per migliorare la loro capacità di generalizzazione. Il dataset utilizzato conteneva sia variabili numeriche che categoriche, rendendo necessarie diverse tecniche di preprocessing per garantire una corretta preparazione dei dati.

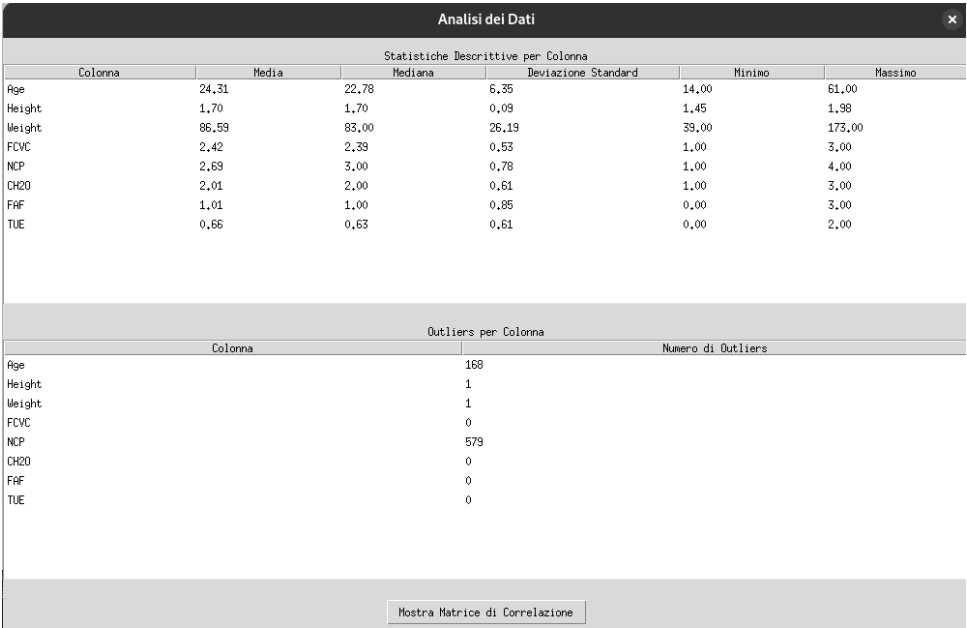


Figura 1: Analisi dei Dati.

## 2 Preprocessing dei Dati

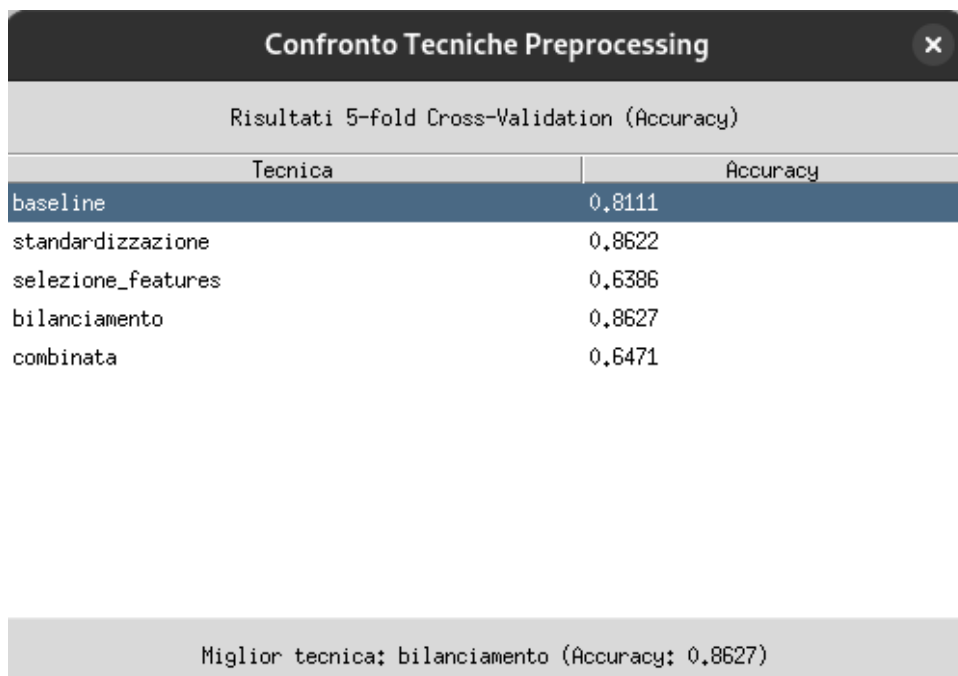
Il preprocessing è stato fondamentale per migliorare la qualità dei dati prima di fornirli ai modelli di apprendimento automatico. Abbiamo iniziato con un'analisi preliminare

(Analisi dei Dati) per identificare eventuali outlier attraverso l'Intervallo Interquartile (IQR) e abbiamo visualizzato la matrice di correlazione per individuare relazioni tra variabili.

Abbiamo creato diverse pipeline utilizzando sia `scikit-learn` che `imblearn`, per mettere a confronto varie strategie di preprocessing. L'idea di base era di analizzare come ogni tecnica, da sola o in combinazione, influenzi le performance del nostro classificatore. Di seguito una panoramica delle pipeline realizzate:

- **Baseline:** utilizzo del dataset "grezzo" senza alcuna trasformazione, per fungere da punto di riferimento.
- **Standardizzazione:** utilizzo dello `StandardScaler` per normalizzare le feature, riducendo le differenze di scala.
- **Selezione delle feature con PCA:** applicazione del PCA configurato per mantenere il 95% della varianza, in modo da ridurre il numero di feature eliminando quelle ridondanti o rumorose.
- **Bilanciamento delle classi:** impiego del `RandomOverSampler` per aumentare la presenza della classe minoritaria e contrastare lo squilibrio del dataset.
- **Pipeline combinata:** integrazione di standardizzazione, PCA e bilanciamento delle classi, per valutare l'effetto complessivo dell'applicazione simultanea di queste tecniche.

Per ogni pipeline è stata utilizzata una cross-validation a 5 fold ( $cv=5$ ) per stimare in maniera robusta l'accuratezza del modello.



Risultati 5-fold Cross-Validation (Accuracy)	
Tecnica	Accuracy
baseline	0.8111
standardizzazione	0.8622
selezione_features	0.6386
bilanciamento	0.8627
combinata	0.6471

Miglior tecnica: bilanciamento (Accuracy: 0.8627)

Figura 2: Tecniche di Preprocessing e relativa accuratezza media.

### 3 Classificatori Implementati

Abbiamo sviluppato diversi modelli di classificazione, sia basati su implementazioni standard che personalizzate:

- **Support Vector Machine (SVM)**: implementata con kernel lineare (tramite `scikit-learn`), particolarmente efficace quando i dati sono (o quasi) linearmente separabili o in presenza di alta dimensionalità. Il kernel lineare, oltre a garantire una buona efficienza computazionale, rende il modello più interpretabile.
- **Decision Tree**: il modello crea una struttura ad albero che suddivide il dataset in base alle feature, fino a giungere alle foglie dove vengono assegnate le classi. Per evitare l'overfitting, si limita la profondità dell'albero.
- **Artificial Neural Network (ANN)**: realizzata con il `MLPClassifier` di `scikit-learn` per costruire una rete neurale multilivello capace di apprendere relazioni non lineari complesse. Sono state testate diverse configurazioni (es. coppie di layer con 10 o 20 neuroni) per massimizzare le prestazioni senza incorrere in overfitting.
- **K-Nearest Neighbors (KNN) Custom**: è stata sviluppata una versione personalizzata del KNN. Il modello supporta diverse metriche di distanza (Manhattan, Euclidea, Chebyshev) e consente di impostare manualmente il numero di vicini ( $k$ ) oppure di ottimizzarlo tramite un processo di tuning.
- **Voting Classifier Custom**: classificatore personalizzato che combina le predizioni di diversi modelli (ad esempio, Decision Tree, KNN e Naive Bayes) tramite voto a maggioranza (Hard Voting). L'idea è quella di sfruttare i punti di forza di ogni modello per ottenere una classificazione più precisa.

### 4 Tuning degli Iperparametri

Per ottenere le migliori prestazioni, abbiamo ottimizzato gli iperparametri dei nostri modelli utilizzando `GridSearchCV` in combinazione con una validazione incrociata a 3 fold. In questo modo, sono state esplorate diverse configurazioni, riducendo il rischio di overfitting e individuando la combinazione ottimale per ciascun classificatore. I parametri testati sono stati:

- **SVM**: ottimizzazione del parametro di regolarizzazione  $C$  con i valori  $\{0.1, 1, 10\}$ .
- **Decision Tree**: ottimizzazione del parametro `max_depth` con i valori  $\{5, 10, 15, 20\}$ .
- **ANN (MLPClassifier)**: variazione del numero di strati nascosti mediante il parametro `hidden_layer_sizes` (es.  $(10, 10)$  e  $(20, 20)$ ).
- **KNN Custom**: tuning del numero di vicini  $k$  (valori testati  $\{3, 5, 7\}$ ) e delle metriche di distanza (Manhattan, Euclidea, Chebyshev).
- **Voting Classifier Custom**: tuning combinato sia del parametro  $k$  per il KNN integrato sia della scelta della metrica di distanza.

L'implementazione nel codice prevede:

- **Uso della Pipeline:** se l'opzione `usePipeline` è attiva, viene creata una pipeline contenente:
  - `StandardScaler` per la standardizzazione,
  - PCA (con `n_components=0.95`) per la riduzione della dimensionalità,
  - `RandomOverSampler` per il bilanciamento delle classi,
  - Il classificatore scelto (`clf`).
- **GridSearchCV:** se il tuning è abilitato (`tuning=True`), si esegue una ricerca a griglia con validazione incrociata a 3 fold, e il miglior valore del parametro viene utilizzato per effettuare le predizioni sul set di test.
- **Visualizzazione dei Parametri Ottimali:** per i modelli KNN e Voting Classifier, se il tuning è attivo, vengono stampati il valore ottimale di  $k$  e la metrica di distanza scelta.

## Tabella Riassuntiva delle Performance

Di seguito sono riportati i valori di accuratezza ottenuti:

Tabella 1: Performance dei Classificatori

Metodo	Classificatori Standard			KNN Custom			Voting Classifier Custom		
	SVM	Decision Tree	ANN	Manhattan	Euclidean	Chebyshev	Manhattan	Euclidean	Chebyshev
<b>Baseline</b>	0.86	0.91	0.94	0.86	0.79	0.70	0.83	0.80	0.75
<b>Tuning</b>	0.94	0.91	0.95	0.86	0.79	0.71	0.85	0.81	0.77
<b>Preprocessing &amp; Tuning</b>	0.78	0.71	0.83	0.79	0.81	0.80	0.77	0.77	0.76

Tabella 2: Confronto tra Tecniche di Preprocessing

Tecnica	Accuratezza
Baseline	0.8111
Standardizzazione	0.8622
Feature Selection (PCA)	0.8386
Bilanciamento (Balance)	0.8627

## 5 Conclusioni

Il progetto ha evidenziato chiaramente l'importanza delle scelte di preprocessing e del tuning degli iperparametri sulle performance dei modelli di classificazione. I punti principali sono:

- **Preprocessing:**
  - L'applicazione della standardizzazione e il bilanciamento delle classi hanno migliorato le performance rispetto all'utilizzo dei dati grezzi (accuratezza di circa 0.86 contro 0.81).

- La combinazione di standardizzazione, PCA e bilanciamento ha portato a un calo di accuratezza (0.65), evidenziando come una riduzione eccessiva della dimensionalità possa eliminare informazioni rilevanti.

- **Tuning degli Iperparametri:**

- L'ottimizzazione dei parametri ha ulteriormente migliorato le performance, ad esempio, l'accuratezza della SVM è passata da 0.86 a 0.94, mentre quella dell'ANN da 0.94 a 0.95.
- Anche per il KNN Custom e il Voting Classifier Custom il tuning ha portato a piccoli miglioramenti.

- **Preprocessing e Tuning Insieme:**

- L'applicazione simultanea delle tecniche di preprocessing e tuning ha portato a una riduzione delle performance (ad esempio, SVM 0.78, Decision Tree 0.71, ANN 0.83), suggerendo che un'eccessiva manipolazione dei dati può avere effetti negativi.

In sintesi, il progetto ha evidenziato come:

- Una corretta analisi dei dati e la scelta adeguata delle tecniche di preprocessing possano fare una grande differenza.
- Il tuning degli iperparametri sia fondamentale per ottenere il massimo dai modelli (specialmente SVM e ANN).
- L'eccesso di operazioni di preprocessing possa risultare controproducente, rendendo necessario trovare il giusto equilibrio.

**Grazie per l'attenzione.**