Classificazione Multiclasse del tipo di vetro

Andrea Terenziani s349167

***Abstract*—Nel settore industriale della produzione e del riciclo del vetro, identificare con precisione la tipologia di vetro è fondamentale per garantire la qualità dei prodotti finali e l'efficienza del processo produttivo.**
Il seguente lavoro affronta il problema della *classificazione automatica* delle tipologie di vetro sulla base della loro *composizione chimica*, analizzando e confrontando sperimentalmente diversi modelli di *machine learning supervisionato*, con il fine di fornire criteri utili alla scelta del metodo più adeguato per le specifiche esigenze aziendali.

I. Introduzione

La classificazione della tipologia di vetro è un passaggio cruciale per l'industria, in particolare per l'integrazione del processo in sistemi automatizzati di produzione e riciclo.
Storicamente, questa attività è stata svolta in laboratorio mediante analisi chimiche e test specifici, procedure affidabili ma caratterizzate da elevati costi operativi e tempi di esecuzione significativi. Negli ultimi anni, l'evoluzione delle tecnologie di *machine learning* ha reso possibile automatizzare il processo di classificazione, riducendo tempi e costi, e aumentando la coerenza delle decisioni.
In questo studio vengono esplorati diversi modelli di apprendimento supervisionato per l'automatizzazione della classificazione del vetro.
Il lavoro analizza nel dettaglio le metodologie seguite, le prestazioni ottenute e la validazione dei modelli secondo differenti metriche.
Poichè i dati da analizzare in questo settore presentano sfide reali come la distribuzione sbilanciata delle classi e la presenza di valori mancanti o anomali, viene inoltre fornita una panoramica completa sul trattamento e sul preprocessing dei dati, con particolare attenzione alle tecniche adottate per garantire l'accuratezza del modello.
Infine, viene fornito un confronto delle prestazioni dei diversi approcci basato sui risultati sperimentali ottenuti, offrendo indicazioni utili per selezionare il metodo più appropriato in base ad obiettivi ed esigenze specifiche di ciascuna azienda.

II. Analisi dei dati

Per lo sviluppo e la valutazione dei modelli sono stati forniti due Dataset distinti, uno destinato all'addestramento e uno al test, contenenti campioni di vetro appartenenti a 7 categorie, descritte da misurazioni numeriche relative alla composizione chimica del materiale e ad alcune variabili derivate.

L'analisi dei dati è stata condotta attraverso ad una sequenza strutturata di fasi, pensate per affrontare in modo sistematico ogni aspetto del problema.
- Analisi Target: come primo passo è stata effettuata un'analisi esplorativa della variabile **target *GlassType*** per valutare la distribuzione delle classi nel dataset di addestramento. L'analisi ha evidenziato uno sbilanciamento significativo: alcune categorie risultano molto più rappresentate di altre e la classe *4:_finestre_veicoli_non_float_elaborate* è **assente**.
- Target Encoding: si è verificato che il target fosse già **codificato numericamente** nei dataset forniti. Pertanto, non è stato necessario applicare alcuna ulteriore codifica tramite *Label Encoding*.

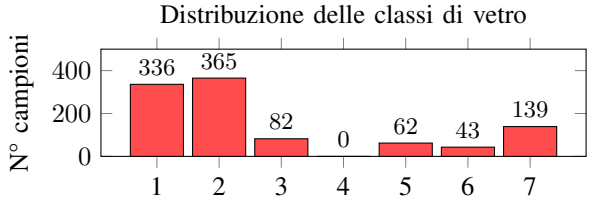

Fig. 1: Distribuzione delle classi di vetro nel dataset di addestramento

- Analisi Features e Preprocessing: è stata condotta un'analisi esplorativa delle *features* con l'obiettivo di valutarne le caratteristiche principali e la qualità dei dati. Sulla base delle osservazioni ottenute, si è proceduto a un *preprocessing* sostanziale, volto a garantire che i dati in ingresso ai modelli fossero coerenti, completi e opportunamente scalati.
- Features Encoding: si è verificato che tutte le features fossero di **tipo numerico**, condizione che ha reso superflua qualsiasi operazione di encoding.
- Ricerca Outliers: sono stati analizzati i *valori anomali* e il rispetto dei *vincoli di dominio*, al fine di individuare eventuali *outliers* presenti nel dataset. Per le variabili chimiche è stato **inizialmente ipotizzato** un vincolo di **non negatività**. L'analisi ha evidenziato la presenza di valori negativi per alcune osservazioni nelle features Mg, K, Ba e Fe. Tuttavia, poiché **oltre il 35%** dei dati presenta valori negativi, si è ipotizzato che essi derivino da unità di misura non documentate. Pertanto, tali campioni sono stati **mantenuti** assumendo valida questa ipotesi.
- Dati Mancanti, Imputazione e Standardizzazione: è stata condotta un'analisi dei *dati mancanti*, rilevando che soltanto due features derivate, *Sodium_Aluminum_Mix* e *Magnesium_Enhanced*, presentavano campioni senza valore.

Per l'imputazione dei valori mancanti è stato adottato il metodo **KNN Imputer**, che completa ciascun campione sulla base dei k esempi più vicini privi di valori nulli, calcolati secondo la distanza euclidea e ponderati in modo inversamente proporzionale alla distanza stessa.

Poiché il metodo KNN si basa sul calcolo delle distanze tra osservazioni, è fondamentale che tutte le feature siano espresse sulla stessa scala. Per questo motivo, prima dell'imputazione, i dati sono stati **standardizzati** tramite lo *StandardScaler*, che applica la trasformazione: $z = \frac{x-\mu}{\sigma}$. Questa operazione elimina distorsioni dovute a scale diverse o unità di misura eterogenee, garantendo che ciascun attributo contribuisca equamente

al calcolo delle distanze.
Per **prevenire** fenomeni di **data leakage**, sia lo *scaler* sia l'imputer sono stati fittati esclusivamente sul *training set*.
- Correlazione Features e Data Reduction: Per valutare possibili ridondanze informative tra variabili, è stata calcolata la **matrice di correlazione**. L'analisi ha evidenziato alcune correlazioni elevate tra coppie di features, suggerendo la possibilità di **ridurre la dimensionalità** del dataset senza perdita significativa di informazione e consentendo, di conseguenza, l'eliminazione di 3 variabili derivate altamente correlate, come mostrato in Fig. 2.
Questa operazione è utile per mitigare il fenomeno della ***curse of dimensionality***: con l'aumentare delle dimensioni, le distanze tra i punti diventano meno significative, compromettendo l'efficacia degli algoritmi basati su metriche di distanza.

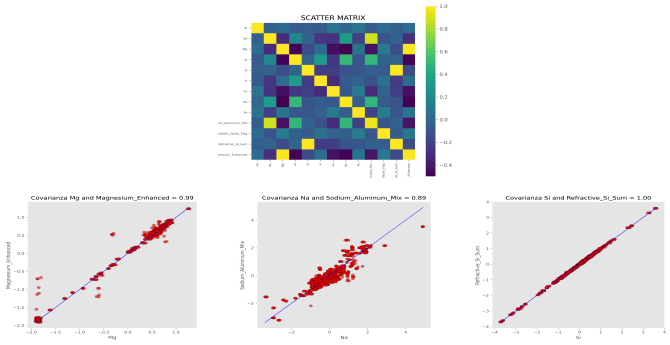

Fig. 2: Analisi grafica della correlazione e delle covarianze tra features (in alto) e confronto della distribuzione di tre coppie di variabili con valore assoluto maggiore (in basso): *Mg* vs *Magnesium_Enhanced* (sinistra), *Na* vs *Sodium_Aluminum_Mix* (centro) e *Si* vs *Refractive_Si_Sum* (destra).

- Principal Component Analysis (PCA): per valutare l'effettiva rilevanza delle fasi di preprocessing, e in particolare della standardizzazione, è stata applicata la *PCA* sia ai dati preprocessati sia a quelli grezzi. La PCA è una tecnica di riduzione della dimensionalità che mira a preservare la varianza originale del dataset, individuando nuove direzioni, dette *componenti principali* (Principal Components, PC), costituite da combinazioni lineari delle variabili originarie, mutualmente ortogonali e ordinate in base alla varianza spiegata.

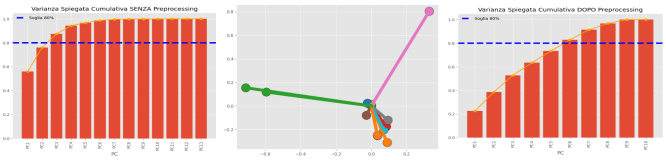

Fig. 3: Confronto grafico tra l'analisi PCA senza preprocessing (sinistra) e con preprocessing (destra) con soglia all' 80% della varianza spiegata totale. Al centro il contributo delle features grezze a PC 1 (ascissa) e PC 2 (ordinata).

I risultati hanno confermato **l'importanza delle operazioni di preprocessing adottate**, evidenziando come, in assenza di esse, i contributi delle singole features siano fortemente influenzati dalla scala di misura e la quasi totalità della varianza sia spiegata da poche features.

III. Metodologia

L'obiettivo del presente studio consiste nello sviluppo di un sistema di classificazione automatica della tipologia di vetro, valutando diversi modelli di apprendimento supervisionato per individuare il modello più adatto alle esigenze industriali. Il flusso di lavoro è stato progettato per garantire coerenza, riproducibilità e capacità di generalizzazione dei modelli. Per assicurare **uniformità nella valutazione**, ciascun classificatore è stato sottoposto alle stesse procedure di preprocessing e alle medesime modalità di validazione, consentendo un confronto coerente tra le diverse soluzioni.

Sono stati selezionati quattro modelli appartenenti a famiglie di algoritmi differenti, così da confrontare approcci con caratteristiche complementari:

- **Decision Tree Classifier** – Modello interpretabile e rapido da addestrare, capace di catturare relazioni non lineari tra le variabili.
- **K-Nearest Neighbors (KNN)** – Algoritmo *lazy learner* che classifica un'osservazione in base alla maggioranza dei k vicini più prossimi secondo una metrica di distanza.
- **Support Vector Machine (SVM)** - Ricerca l'iperpiano di separazione ottimale tra le classi e, attraverso kernel non lineari, gestisce efficacemente dati non linearmente separabili.
- **Multi-Layer Perceptron (MLP)** – Rete neurale feed-forward in grado di apprendere relazioni complesse attraverso l'ottimizzazione iterativa dei pesi.

Poiché ciascun algoritmo richiede l'ottimizzazione di specifici iperparametri, la selezione della configurazione ottimale è stata effettuata mediante ricerca su griglia, **Grid-Search**, combinata con validazione incrociata a k-fold. Come criterio di scelta è stato adottato il valore massimo di ***F1-macro score***[1]ottenuto sul set di validazione. Terminata la fase di addestramento, le prestazioni dei modelli sono state analizzate mediante diversi indicatori, per valutarne in dettaglio i risultati e approfondire vari aspetti di interesse.
- F1-macro score: l'indicatore principale adottato che, combinando in un unico valore sia *precision* che *recall*, fornisce una misura bilanciata e più adatta della *accuracy* nel contesto multiclasse e sbilanciato di questo studio.
- Confusion Matrix: matrice che riassume predizioni corrette ed errori (*falsi positivi* e *falsi negativi*) per ciascuna classe, permettendo di valutare in modo dettagliato la qualità del modello e individuare eventuali errori sistematici di classificazione.
- Learning Curve: curva che mostra l'andamento delle prestazioni e la capacità di apprendimento del modello al variare della dimensione del training set
- Analisi Overfitting/Underfitting: confronto tra le prestazioni su training e test set, supportato dallo studio dell'andamento dell'errore percentuale al variare della complessità del modello
- ROC[2] **e AUC**: la curva *ROC* e il relativo *AUC* (*Area Under the Curve*) forniscono una misura sintetica e complessiva delle prestazioni e della capacità discriminativa del modello.

[1]**F1-macro score**: Media aritmetica degli *F1-score* calcolati su ciascuna classe. L'*F1-score* è definito come $F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

[2]**ROC (Receiver Operating Characteristic)**: rappresentazione grafica della capacità di un classificatore di distinguere tra classi, ottenuta tracciando il tasso di veri positivi contro il tasso di falsi positivi al variare della soglia di decisione.

Di seguito si riportano le principali caratteristiche operative e gli iperparametri considerati per ciascun modello durante la fase di ottimizzazione.

- **Decision Tree (DT)** – È stato adottato l'algoritmo *CART*, che seleziona **dinamicamente** la feature e la soglia di split ottimali secondo un criterio di impurità (*Gini* o *Entropia*). La *Grid Search* ha esplorato diverse combinazioni di iperparametri, variando il criterio di split, la profondità massima dell'albero, il numero minimo di campioni per split e la soglia minima di riduzione dell'impurità.
- **K-Nearest Neighbors (KNN)** – Essendo sensibile alla scala delle feature, è stata applicata la **standardizzazione** descritta nella Sezione II. La *Grid Search* ha variato il numero di vicini considerati, la metrica di distanza, lo schema di pesatura dei vicini e l'algoritmo di ricerca.
- **Support Vector Machine (SVM)** – Modello di classificazione originariamente binario. Nel caso multiclasse, è necessario considerare uno dei seguenti approcci: l'approccio **One-vs-Rest (OvR)**, con C classificatori binari ciascuno contro tutte le altre classi, e quello **One-vs-One (OvO)**, con $C(C-1)/2$ classificatori addestrati su coppie di classi. Oltre a tali strategie, sono stati analizzati come iperparametri il parametro di penalizzazione C e il tipo di kernel utilizzato per proiettare i dati in uno spazio a dimensionalità superiore.
- **Multi-Layer Perceptron (MLP)** – Rete neurale feed-forward ottimizzata mediante **backpropagation**. Gli iperparametri ottimizzati sono: numero e dimensioni layers, activation function, ottimizzatore, L_2 e numero iterazioni.

IV. Risultati sperimentali

Tutti i modelli descritti nella Sezione III sono stati sottoposti all'insieme di analisi illustrate in precedenza, sia sul *train Dataset* che sul *test Dataset*, condotte in maniera uniforme per ciascun classificatore, al fine di garantire un confronto equo e coerente tra le diverse architetture.

In questa sezione vengono tuttavia riportati soltanto i risultati più significativi, scelti per la loro rilevanza ai fini del confronto finale e della valutazione dell'efficacia dei modelli. Per ogni metodo sono quindi presentati:

- la **confusion matrix** delle predizioni sul *test set*;
- il valore di **F1-macro score** calcolato sul *test set*;
- la **combinazione ottimale di iperparametri** individuata tramite *Grid Search* con validazione incrociata;
- eventuali ulteriori evidenze sperimentali ritenute rappresentative delle prestazioni e del comportamento del modello.

Le prestazioni dei modelli sono descritte e commentate separatamente, seguite da una sintesi comparativa che mette in evidenza punti di forza e limiti di ciascun approccio.

A. *Decision Tree*

Iperparametri ottimali: {criterion: 'entropy', max_depth: 30, min_impurity_decrease: 0.005, min_samples_split: 6}

Uno degli aspetti di maggiore interesse per questo modello è la sua interpretabilità: il primo nodo di splitting dell'albero corrisponde alla feature **Mg**, in quanto massimizza la riduzione di impurità (entropia).

Pred / True	1	2	3	4	5	6	7
1	79	3	2	0	0	0	0
2	1	90	0	0	0	0	0
3	2	0	18	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	16	0	0
6	0	0	0	0	0	11	0
7	0	0	0	0	0	0	35

Decision Tree
Confusion matrix
F1-macro = 0.972

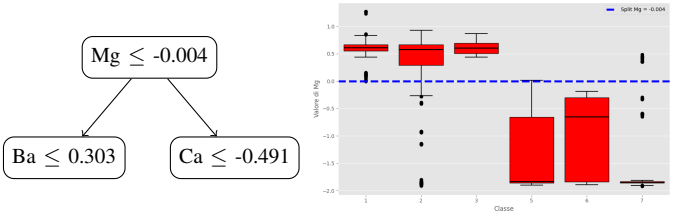

Fig. 4: Primo split dell'albero decisionale (a sinistra) e Box Plot della distribuzione delle classi target rispetto alla feature Mg (a destra).

Tale comportamento è coerente con quanto osservabile nel **box plot** in *Fig. 4*, che mostra la distribuzione della variabile *Mg* per ciascuna classe target: la soglia selezionata dall'albero è in grado di separare le distribuzioni in modo netto minimizzando l'overlap tra di esse.

B. *K-Nearest Neighbors (KNN)*

Iperparametri ottimali: {algorithm: 'auto', metric: 'manhattan', n_neighbors: 2, weights: 'distance'}

Pred / True	1	2	3	4	5	6	7
1	81	1	1	0	0	0	1
2	2	88	1	0	0	0	0
3	2	2	16	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	16	0	0
6	0	0	0	0	0	11	0
7	0	0	0	0	0	0	35

K-Nearest Neighbors
Confusion matrix
F1-macro = 0.959

Le analisi hanno evidenziato un comportamento significativo della **learning curve**, utilizzata per osservare l'evoluzione delle prestazioni in funzione della dimensione del training set. Il comportamento osservato, con un punteggio perfetto sul set di addestramento, **non è indice di *overfitting*** ma è conseguenza naturale del funzionamento dell'algoritmo: utilizzando pesi proporzionali all'inverso della distanza (`weights='distance'`), ogni campione **del training set** assegna a sé stesso la massima probabilità di appartenenza, risultando quindi classificato correttamente.

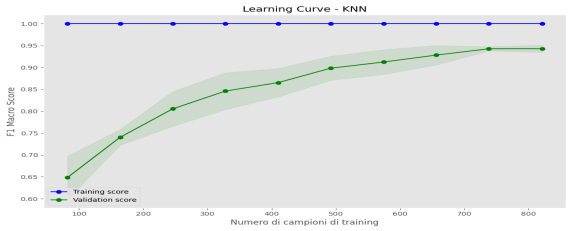

Fig. 5: Learning curve-KNN: mostra l'*F1-macro score* all'aumentare del numero di campioni utilizzati per il train.

C. *Support Vector Machine (SVM)*

Iperparametri ottimali: {C:10,decision_function_shape:'ovo', gamma:'scale', kernel:'rbf'}

Pred / True	1	2	3	4	5	6	7
1	78	6	0	0	0	0	0
2	10	81	0	0	0	0	0
3	5	2	13	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	16	0	0
6	0	0	0	0	0	11	0
7	2	0	0	0	0	0	33

Support Vector Machine
Confusion matrix
F1-macro = 0.922

Per fornire una rappresentazione visiva del comportamento del modello, è stata realizzata una **proiezione bidimensionale** delle decisioni del classificatore. I dati di test sono prima stati predetti, e successivamente ridotti mediante *PCA* su due componenti principali per poter essere proiettati nello spazio 2D. Si precisa che tale rappresentazione è **approssimativa** in quanto la riduzione dimensionale **non è un'operazione biettiva** e quindi non è invertibile.

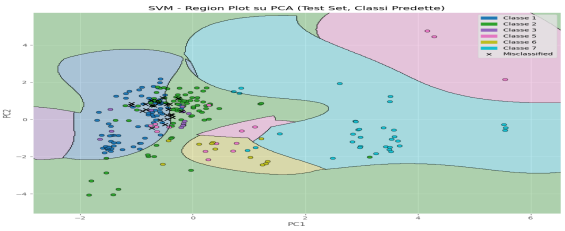

Fig. 6: Campioni e Regioni di decisione SVM, proiettati in 2D tramite PCA.

D. *Multi-Layer Perceptron (MLP)*

Iperparametri ottimali:
activation:'relu', hidden_layer_sizes:(30,30), max_iter:3000

Pred / True	1	2	3	4	5	6	7
1	82	1	1	0	0	0	0
2	3	88	0	0	0	0	0
3	1	0	19	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	16	0	0
6	0	0	0	0	0	11	0
7	1	0	0	0	0	0	34

MLP
Confusion matrix
F1-macro = 0.979

Tra i diversi esperimenti condotti per analizzare fenomeni di **overfitting** e **underfitting** nei modelli MLP, una configurazione particolarmente significativa analizzata è stata l'influenza della *depth* mantenendo il numero di neuroni fissi.

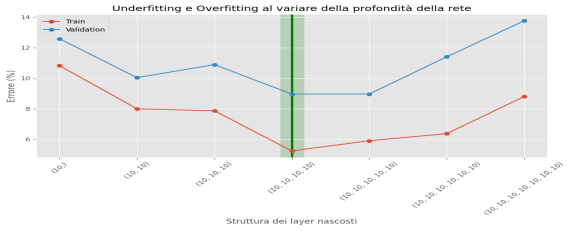

Fig. 7: Andamento delle prestazioni dell'MLP al variare del numero di layer, mantenendo costante il numero di neuroni per layer.

Come si osserva nel grafico, compare un evidente **overfitting** dopo circa 4 layer, con un progressivo peggioramento delle prestazioni sia sul set di addestramento che di test.

E. *Confronto Modelli*

Innanzitutto, è stata condotta un'analisi approfondita delle **curve ROC** per ciascun modello. Tali curve sono state utilizzate anche per confrontare le prestazioni ottenute con i **dati grezzi** rispetto a quelli **preprocessati**, evidenziando come le operazioni di pretrattamento abbiano avuto un impatto significativamente positivo sulla separabilità delle classi. In particolare, il modello MLP — che si è rivelato il più performante — ha mostrato un **incremento dell'11.1%** in termini di AUC **a seguito del preprocessing**. Poiché in tutti i modelli l'AUC risulta prossima a 1, è stato eseguito uno **zoom** sulle porzioni più significative del grafico

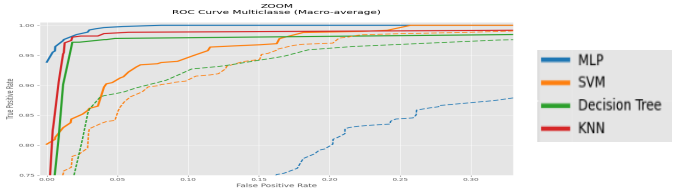

Fig. 8: Curve ROC, dove i modelli applicati a dati grezzi sono tratteggiati. Lo zoom evidenzia l'intervallo compreso tra [0, 0.35] per il *False Positive Rate* (ascissa) e [0.75, 1] per il *True Positive Rate* (TPR) (ordinata)

Infine, è stato effettuato un **confronto tra tutte le metriche** di valutazione per ciascun modello sul test set, evidenziando i valori migliori per ciascuna metrica.

Metrica	DT	KNN	SVM	MLP
Accuracy	0.969	0.961	0.903	0.973
Recall	0.972	0.964	0.955	0.980
Precision	0.972	0.955	0.902	0.977
F1-score	0.972	0.959	0.922	0.979
AUC	0.99	0.99	0.99	1.00

V. Conclusioni

Tutti i modelli testati hanno mostrato buone prestazioni di classificazione, confermando la validità dell'approccio. È emersa in particolare l'**importanza cruciale del preprocessing** dei dati: la sua corretta esecuzione ha permesso di migliorare significativamente l'efficacia delle classificazioni per ogni modello.

Tra i modelli analizzati, l'**MLP** si è dimostrato il più performante, risultando superiore in tutte le metriche considerate, e rappresenta quindi, in questo contesto, la scelta consigliata per scenari in cui si desidera la massima accuratezza.

Tuttavia, anche gli altri modelli possono essere adottati in funzione delle esigenze specifiche di un'azienda. Ad esempio, se è fondamentale ridurre al minimo i *falsi positivi* (ovvero ottenere un basso **False Positive Rate, FPR**), i modelli **KNN** e **Decision Tree** risultano essere una valida opzione.

Al contrario, se l'obiettivo è di privilegiare la **sensibilità su classi specifiche**, anche a costo di accettare un FPR più elevato, l'**SVM** risulta particolarmente indicato: per le **classi 3, 4, 5, 6 e 7** è infatti l'unico modello a garantire una **recall perfetta**, aspetto cruciale per **riconoscere correttamente tutti** gli elementi appartenenti a tali classi.

Nel complesso, il processo di classificazione automatica del vetro risulta tecnicamente realizzabile con un elevato grado di affidabilità, e rappresenta una concreta opportunità di automazione per applicazioni industriali o analitiche.

VI. Codice e Riproducibilità

Gli script utilizzati sono disponibili pubblicamente al link:
https://github.com/terenziani/Glass_Classification_BIXBD