

# Integrazione di conoscenza di dominio e apprendimento automatico per la previsione della qualità del vino

Gruppo di lavoro

Giuseppe Murano, 758407, g.murano2@studenti.uniba.it

[https://github.com/P3pp0uz/Progetto\\_ICON\\_2024\\_2025\\_Giuseppe\\_Murano.git](https://github.com/P3pp0uz/Progetto_ICON_2024_2025_Giuseppe_Murano.git)

AA 2024-2025

## Sommario

Introduzione.....	3
Sommario.....	3
Dataset .....	3
Apprendimento non supervisionato.....	6
Apprendimento supervisionato.....	10
Ragionamento probabilistico.....	16
Rappresentazione della Conoscenza .....	19
Conclusioni .....	21
Requisiti Tecnici e Riproducibilità .....	22
Riferimenti Bibliografici .....	22

# Introduzione

Il progetto affronta la predizione della qualità del vino a partire da misure fisico-chimiche su un dataset UCI unificato per rossi e bianchi, integrando conoscenza di dominio, modelli probabilistici e apprendimento supervisionato e non supervisionato in un'unica pipeline eseguibile da main.py. L'obiettivo è dimostrare competenze pratiche "in programma" su rappresentazione della conoscenza e ragionamento, apprendimento e valutazione rigorosa, con particolare enfasi sulla KB e sulle decisioni motivate di configurazione dei modelli.

## Sommario

Il sistema integra cinque diversi moduli che dimostrano competenze nei seguenti argomenti del corso:

- 1) Una **Knowledge Base (KB) basata su regole** per l'inferenza della qualità dei vini, utilizzando il paradigma di forward chaining descritto nel Capitolo 13: Ontologie e Knowledge-Based Systems del libro.
- 2) Una **pipeline di apprendimento supervisionato multi-algoritmo** direttamente collegata ai contenuti del Capitolo 7 ("Learning Overview and Supervised Learning").
- 3) Un **MLP (Multilayer Perceptron) regolarizzato** secondo i modelli descritti nel **Capitolo 7.3.2** (Linear Regression and Classification) e il Capitolo 8 sulle reti neurali.
- 4) Dei **modelli probabilistici** grafici per la classificazione descritti nel Capitolo 6 ("Reasoning Under Uncertainty").
- 5) Implementa **clustering non supervisionato** con K-Means e selezione automatica del numero di cluster, capitolo 10.

## Dataset

Nel presente caso di studio si è scelto di utilizzare il **Wine Quality dataset**, un insieme di dati ampiamente riconosciuto nel campo dell'analisi enologica e della valutazione qualitativa dei vini. Questo dataset contiene 6497 istanze complessive estratte da campioni reali di "Vinho Verde", un vino tipico del nord del Portogallo, le quali descrivono le caratteristiche chimico-fisiche derivanti da test di laboratorio oggettivi.

### Composizione Dataset

Il dataset inizialmente era suddiviso in due varianti:

- **Vini rossi** (winequality-red.csv): 1599 campioni
- **Vini bianchi** (winequality-white.csv): 4898 campioni

Entrambi i sottoinsiemi condividono la stessa struttura di 11 attributi di input chimico-fisici e 1 attributo di output qualitativo basato su valutazioni sensoriali di esperti enologi per questo si è deciso di unirli in un unico dataset.

## Variabili di Input (Feature Chimico-Fisiche)

Tutte le feature sono di tipo **continuo** e derivano da analisi di laboratorio standardizzate:

1. **Fixed Acidity** (Acidità fissa): concentrazione di acidi non volatili (es. tartarico, malico, citrico) espressa in g/dm<sup>3</sup>
2. **Volatile Acidity** (Acidità volatile): concentrazione di acido acetico, indicatore di fermentazione indesiderata se elevata, espressa in g/dm<sup>3</sup>
3. **Citric Acid** (Acido citrico): concentrazione di acido citrico, che contribuisce alla freschezza del vino, espressa in g/dm<sup>3</sup>
4. **Residual Sugar** (Zucchero residuo): quantità di zucchero rimanente dopo la fermentazione, espressa in g/dm<sup>3</sup>
5. **Chlorides** (Cloruri): concentrazione di sale, espressa in g/dm<sup>3</sup>
6. **Free Sulfur Dioxide** (Anidride solforosa libera): forma libera di SO<sub>2</sub>, agente antimicrobico e antiossidante, espressa in mg/dm<sup>3</sup>
7. **Total Sulfur Dioxide** (Anidride solforosa totale): somma delle forme libere e legate di SO<sub>2</sub>, espressa in mg/dm<sup>3</sup>
8. **Density** (Densità): massa volumica del vino, correlata al contenuto alcolico e zuccherino, espressa in g/cm<sup>3</sup>
9. **pH**: misura dell'acidità/alcalinità su scala logaritmica
10. **Sulphates** (Solfati): concentrazione di solfato di potassio, additivo antimicrobico, espressa in g/dm<sup>3</sup>
11. **Alcohol** (Alcol): percentuale di alcol etilico in volume (% vol)

## Variabile di Output (Target)

12. **Quality** (Qualità): punteggio qualitativo assegnato da esperti enologi su scala discreta da 0 a 10, basato su valutazione sensoriale (colore, aroma, gusto)
  - **Distribuzione sbilanciata**: la maggior parte dei campioni ha qualità medio-bassa (5-6), con pochi vini eccellenti (8-9) o scarsi (3-4)
  - **Classificazione binaria**: nel progetto, la qualità viene binarizzata con soglia  $\geq 6$  per distinguere "Alta qualità" (1) da "Bassa qualità" (0)

## Caratteristiche del Dataset

- **Dominio reale**: i dati provengono da campioni fisici sottoposti ad analisi chimiche certificate secondo standard CVRVV (Commissione per la Viticoltura della Regione del Vinho Verde)
- **Task supportati**: classificazione (binaria o multi-classe), regressione, rilevamento outlier per vini eccezionali
- **Privacy**: per motivi di riservatezza commerciale, il dataset non include informazioni su vitigno, marca, prezzo di vendita o origine geografica specifica

## Fase di Preprocessing

La funzione preprocess\_data () si occupa di preparare il dataset per l'analisi attraverso quattro operazioni principali:

**1. Normalizzazione dei nomi delle colonne:** I nomi delle colonne vengono ripuliti rimuovendo spazi bianchi iniziali e finali (strip ()) e identificando pattern di colonne non valide (e.g., Unnamed:0). Questa fase garantisce consistenza nella nomenclatura, fondamentale per l'integrazione con la knowledge base enologica.

**2. Codifica del tipo di vino:** Se presente, la colonna categorica type viene mappata numericamente ('red' → 0, 'white' → 1) per consentire l'elaborazione da parte degli algoritmi di machine learning. Questa trasformazione preserva l'informazione tipologica in formato compatibile con i modelli matematici.

**3. Conversione numerica robusta:** Tutte le 11 feature chimico-fisiche note del dataset UCI (fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol) e la variabile target quality vengono esplicitamente convertite a tipo numerico tramite pd.to\_numeric () con parametro errors='coerce'. Questa strategia gestisce eventuali errori di parsing convertendo valori non parsabili in NaN, evitando interruzioni del flusso di elaborazione.

**4. Gestione valori mancanti:** Sebbene il dataset UCI Wine Quality ufficiale non contenga valori mancanti, il sistema implementa una verifica precauzionale rimuovendo istanze con NaN solo sulle 11 feature chimico-fisiche critiche, senza eliminare l'intero DataFrame. Questo approccio selettivo preserva la massima quantità di informazione disponibile.

## Binarizzazione del Target

La funzione make\_binary\_target () converte la variabile qualitativa ordinale quality (scala 0-10) in un target binario per classificazione:

- **Alta qualità** (1): vini con punteggio  $\geq 6$
- **Bassa qualità** (0): vini con punteggio  $< 6$

La soglia di 6 è stata scelta analizzando la distribuzione sbilanciata del dataset, dove la maggior parte dei campioni si concentra nei valori 5-6. Il sistema include un rilevamento automatico intelligente della colonna qualità tramite pattern matching su nomi candidati (quality, Quality, quality\_score) e analisi euristica per identificare colonne con valori discreti nell'intervallo.

# Apprendimento non supervisionato

## Sommario

Il modulo **unsupervised learning** implementa una pipeline di clustering K-Means per scoprire strutture naturali nei dati dei vini senza etichette predefinite. L'obiettivo è identificare gruppi omogenei di vini basati su similarità delle proprietà chimico-fisiche, quindi estrarre feature semantiche aggiuntive (etichette cluster, one-hot encoding, distanze dai centri) che arricchiscono il dataset originale per le fasi successive di apprendimento supervisionato e reti neurali.

La pipeline esegue sequenzialmente: normalizzazione MinMaxScaler dei dati, calcolo della **Within Sum of Squares (WSS)** per un range di k valori (2-10), applicazione **dell'euristica del gomito** per selezione automatica di k ottimale, clustering K-Means effettivo, estrazione di tre rappresentazioni feature, visualizzazione mediante PCA 2D dei cluster e grafico elbow.

## Strumenti utilizzati

### Librerie Principali

scikit-learn fornisce tre componenti essenziali: KMeans per clustering, PCA per riduzione dimensionalità e visualizzazione, MinMaxScaler per normalizzazione.

Numpy gestisce operazioni matriciali,

Matplotlib genera visualizzazioni.

## Decisioni di Progetto

### Normalizzazione MinMaxScaler

I dati vengono scalati nel range anziché standardizzati (media 0, varianza 1) per un motivo specifico: il calcolo della distanza euclidea in K-Means è sensibile alla scala delle feature. MinMaxScaler garantisce che tutte le feature (alcolici, acidi, densità, etc.) contribuiscano equamente al calcolo della distanza euclidea, evitando che feature con range più grande (es. solidi disciolti totali) dominino il clustering.

### Range di k e Convergenza

Il range 2-10 per k è scelto perché rappresenta il compromesso tra semplicità interpretativa (k troppo grande frammenta i dati) e capacità di catturare varietà naturale nei vini (k=2 potrebbe essere troppo semplicistico). Durante il calcolo WSS, n\_init=20 (multipli tentativi) e max\_iter=500 garantiscono convergenza robusta di K-Means anche per k grandi.

### Estrazione Triplice di Feature

Dopo clustering, vengono estratte tre rappresentazioni alternative dal medesimo modello K-Means, catturando aspetti complementari della struttura scoperta:

- **Etichette cluster (1 colonna)**: identificazione diretta del cluster, variabile categorica discreta, codifica l'appartenenza di massa
- **One-hot encoding (k colonne)**: rappresentazione sparsa con 1 nella posizione del cluster, 0 altrove, fornisce feature binarie esplicite per modelli lineari
- **Distanze dai centri (1 colonna)**: quanto "periferico" è il punto nel suo cluster, cattura irregolarità geometriche della struttura, valori alti = punti al confine tra cluster

Questa triplice estrazione massimizza l'informazione semantica disponibile per algoritmi supervisionati successivi, poiché ogni rappresentazione cattura aspetti diversi della struttura trovata.

Nel main.py, clustering non supervisionato viene eseguito **prima** di apprendimento supervisionato, permettendo che le feature estratte arricchiscano il dataset originale prima dell'addestramento dei modelli.

## Valutazione

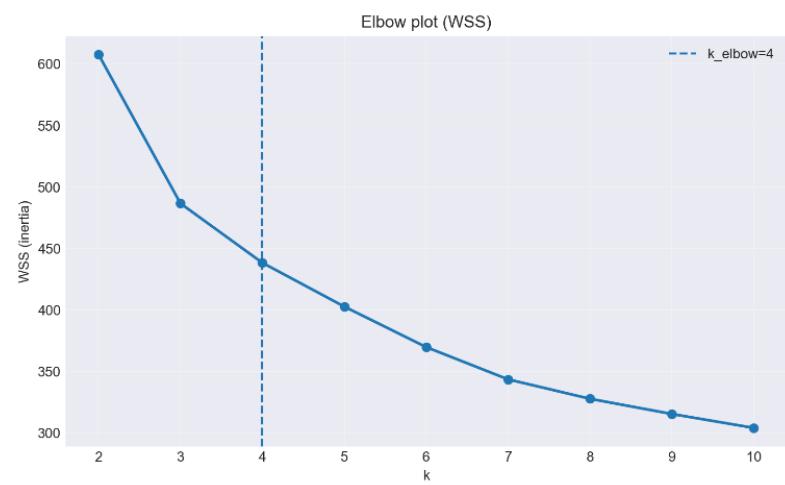
### Metriche

**Within Sum of Squares (WSS)**: WSS diminuisce monotonicamente con k crescente (aggiungere cluster riduce sempre WSS), ma la velocità di diminuzione rallenta dopo il k ottimale.

**Punto gomito**: identificato da `_elbow_k()` come il valore di k con massima distanza perpendicolare dal segmento che connette  $(2, WSS_2)$  a  $(10, WSS_{10})$ . Rappresenta il trade-off ottimale tra semplicità (pochi cluster) e fit (WSS basso).

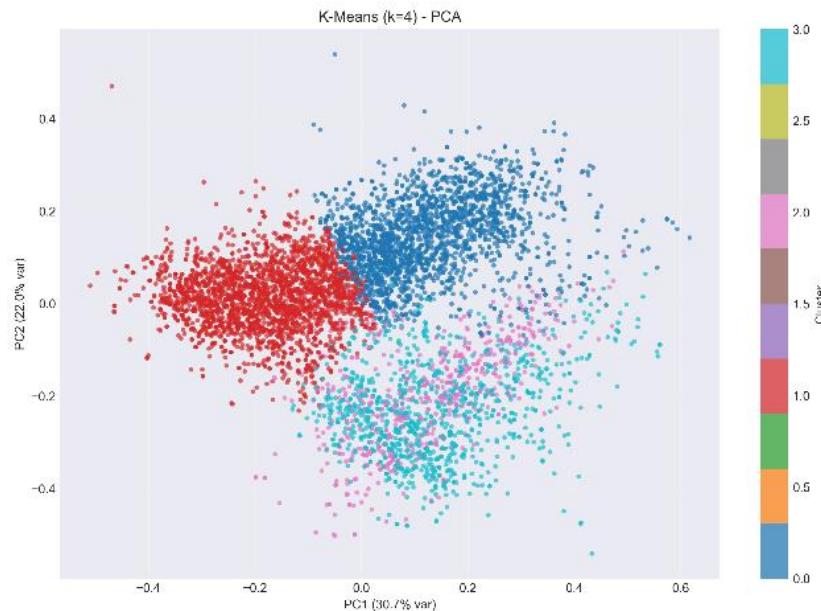
### Visualizzazioni Generate

**Elbow Plot**: Grafico lineare WSS vs k con linea verticale al k\_elbow scelto. Permette verifica visiva della scelta automatica e identificazione manuale di gomiti alternativi se la curvatura è ambigua.

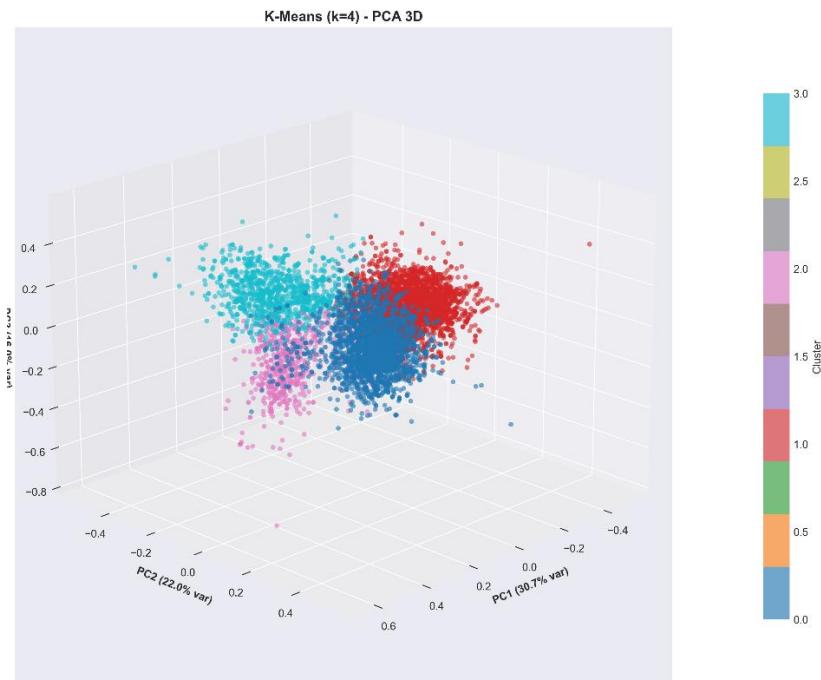


**Clustering Visualization 2D via PCA**: Scatter plot che proietta i dati sui primi due principali componenti ( $PC_1$  e  $PC_2$ ), con colori differenziati per cluster assegnato. Rivela la struttura

geometrica effettiva del clustering: cluster ben separati appaiono come regioni spaziali distinte, cluster sovrapposti appaiono intrecciati.

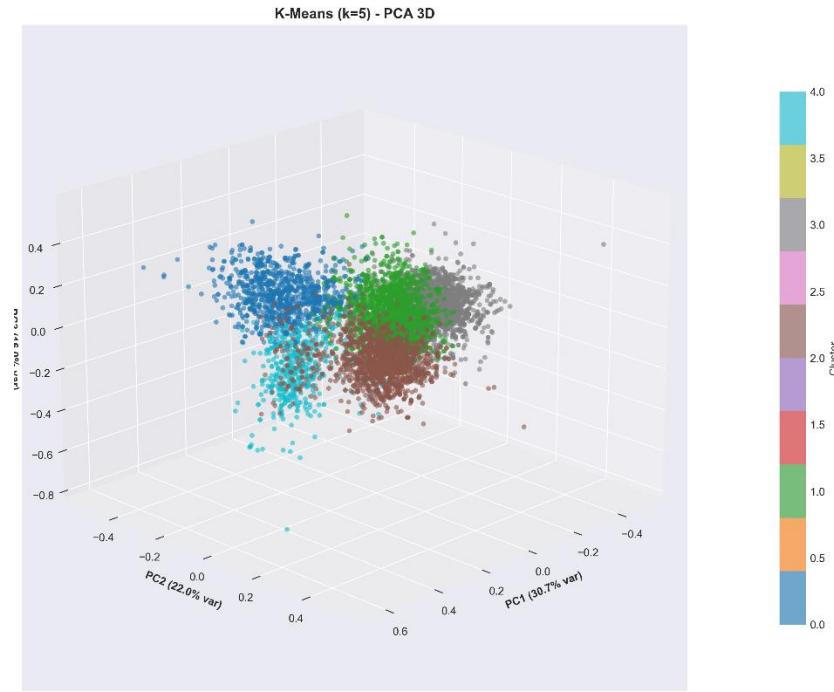


La visualizzazione PCA 2D iniziale (scatter plot su PC1 e PC2) rivelava un'ambiguità strutturale critica: i cluster etichettati in rosa (cluster 2) e blu (cluster 3) risultavano spazialmente sovrapposti o contigui nel piano PC1-PC2, rendendo incerta la separazione effettiva fra questi due gruppi. In altre parole, la proiezione su soli due assi principali comprimeva l'informazione 3D dello spazio feature originale, mascherando potenziali divisioni geometriche che esistevano nelle dimensioni non visualizzate.



Per risolvere questa ambiguità, è stata generata una visualizzazione PCA 3D che proietta i dati sulle prime tre componenti principali (PC1, PC2, PC3). La terza dimensione cattura ulteriore varianza e rivela la struttura geometrica nascosta del clustering.

**Risultato della PCA 3D:** i cluster precedentemente sovrapposti in 2D risultano **chiaramente** separati nello spazio 3D, con regioni spaziali distinte e non-intersecanti. Questo indica che la confusione era un artefatto di proiezione, non una caratteristica reale dei dati. La visualizzazione 3D conferma che i cluster sono genuinamente discreti e geometricamente separabili, validando l'approccio di clustering.



### Validazione con Test Forzato k=5

Per confermare ulteriormente la scelta di  $k$ , è stato eseguito un test esplicito con  $k=5$  cluster **forzato** (indipendentemente dal  $k_{\text{elbow}}$  trovato dall'algoritmo automatico). L'analisi PCA 3D del clustering con  $k=5$  ha rivelato che:

- Quattro cluster risultano ben separati e coesi nello spazio 3D, con centroidi distinti
- Il quinto cluster emerge come artificiale: rappresenta una frammentazione minore di uno dei 4 cluster principali, senza supporto geometrico robusto

Questo risultato **valida la scelta di  $k=4$**  come numero ottimale: rappresenta il compromesso naturale fra semplicità interpretativa e fedeltà strutturale ai dati. Con  $k=4$ , ogni cluster corrisponde a una regione geometrica distinta nello spazio enologico, probabilmente riflettendo sottopopolazioni naturali di vini (es. rossi di qualità alta/bassa, bianchi di qualità alta/bassa).

# Apprendimento supervisionato

## Sommario

La pipeline completa di classificazione combina due paradigmi complementari di machine learning per la predizione della qualità dei vini (alta/bassa).

L'apprendimento Supervisionato implementa una comparazione di 5 algoritmi classici (KNN, Decision Tree, Random Forest ottimizzato, Gradient Boosting, SVM) con regolarizzazione anti-overfitting, eseguendo feature selection, normalizzazione, ottimizzazione iperparametri con RandomizedSearchCV, e cross-validation stratificata 5-fold.

Le reti neurali implementano un'architettura MLP leggera (64-32 hidden layers) adatta a dataset di piccola-media dimensione (~6000 campioni), con early stopping per prevenire overfitting, regolarizzazione L2 (alpha=0.03), e learning rate adattivo. Entrambi i moduli utilizzano lo stesso dataset, split stratificato 85-15%, metriche comparative standardizzate (Accuracy, Precision, Recall, F1, MCC) e visualizzazioni unificate per decidere quale approccio è più efficace.

## Strumenti utilizzati

Entrambi i moduli utilizzano scikit-learn: MLPClassifier per le reti neurali, KNeighborsClassifier, DecisionTreeClassifier, RandomForestClassifier, GradientBoostingClassifier, SVC per apprendimento supervisionato.

StandardScaler normalizza i dati in entrambi gli approcci.

Intel Extension for Scikit-Learn (sklearnex) fornisce accelerazione computazionale.

## Architettura MLP

La rete neurale artificiale implementa una **topologia Multilayer Perceptron** con backpropagation:

- **Input layer:** numero feature pari alla dimensionalità del dataset processato.
- **Hidden layer 1:** 64 neuroni con attivazione ReLU per non-linearità.
- **Hidden layer 2:** 32 neuroni con attivazione ReLU per rappresentazioni apprese complesse.
- **Output layer:** 2 neuroni (probabilità classi binarie).
- **Solver:** Adam (adaptive moment estimation) per ottimizzazione del gradiente.

## Regolarizzazione nella Rete

Regolarizzazione L2 (alpha=0.03) penalizza i pesi grandi durante il training, prevenendo sovraadattamento. Early stopping monitora un validation set del 20% e interrompe il training se non ci sono miglioramenti per 15 iterazioni consecutive. Batch size 64 equilibra stabilità e velocità computazionale.

## **Decisioni di Progetto**

### **Comparazione Metodologica**

La scelta di unificare il preprocessing fra i due moduli (supervisionato e reti neurali) risponde a un principio fondamentale della valutazione empirica comparativa: quando diversi paradigmi di apprendimento lavorano su rappresentazioni identiche, le differenze prestazionali riflettono genuini trade-off algoritmici piuttosto che artefatti di preprocessing.

Lo split stratificato 85-15% preserva la distribuzione di classe nel train e test, essenziale con dati leggermente sbilanciati (53% vs 47%). La feature selection con SelectKBest k=19 riduce lo spazio ipotesi mantenendo solo feature discriminative, diminuendo il rischio di overfitting su dataset di taglia media (~6000 campioni). StandardScaler normalizza l'intervallo dei valori, requisito critico per algoritmi sensibili alla scala (SVM, MLP, KNN) dove distanze euclidee e gradienti di backpropagation dipendono dalla magnitudine assoluta delle feature.

### **Configurazione KNN**

Il choice n\_neighbors=5 bilancia due forme complementari di errore: con k troppo piccolo (k=1 o k=3), il modello memorizza il rumore locale dei singoli vicini isolati, causando varianza alta; con k troppo grande (k=20), il modello sovrasemplifica le strutture locali, causando bias alto e perdita di granularità decisionale. Per dataset di ~6000 campioni con 19 feature, k=5 rappresenta empiricamente un punto di equilibrio dove la "votazione locale" cattura pattern genuini senza contaminarsi da outlier.

L'assegnazione weights='uniform' è conservativa perché i dati enologici non presentano clustering fortemente euclideo: le distanze metriche non seguono geometrie regolari dove punto vicino → necessariamente più importante. L'alternativa weights='distance' darebbe sovra-enfasi a outlier chimici isolati che casualmente giacciono molto vicini, rischiando previsioni erratiche su nuovi campioni.

### **Decision Tree Singolo**

Il max\_depth=8 rappresenta un freno esplicito contro la crescita incontrollata dell'albero: senza questo vincolo, l'albero crescerrebbe fino a profondità elevata, creando troppe foglie potenziali e memorizzando rumore nel training set. Una profondità di 8 produce al massimo ~256 nodi, sufficienti a stratificare ricorsivamente 19 feature in 8 livelli di decisioni binarie, ma ancora gestibile in termini di interpretabilità e generalizzazione.

I vincoli min\_samples\_split=15 e min\_samples\_leaf=7 impongono un filtro demografico sui nodi: ogni foglia deve contenere almeno 7 campioni, evitando che il modello crei previsioni su minuscoli sottogruppi di training che sono verosimilmente rumore. La relazione empirica  $\text{min\_samples\_leaf} \approx \text{min\_samples\_split} / 2$  assicura proporzionalità fra il numero di campioni necessari per uno split e il numero residuo dopo lo split, mantenendo coerenza strutturale.

La scelta max\_features='sqrt' riduce la correlazione fra split successivi sull'albero: ad ogni nodo, l'algoritmo considera solo  $\sqrt{19} \approx 4$  feature casuali anziché tutte le 19, introducendo decorrelazione. Questo meccanismo è cruciale in Random Forest (dove lo stesso effetto su k alberi costruisce ensemble robusto) e utile anche per il singolo albero perché forza la ricerca di split alternativi, evitando che l'albero si ancore a una singola feature dominante.

## Random Forest Ottimizzato

L'uso di RandomizedSearchCV con 50 iterazioni è pragmatico: una GridSearch esaustiva su 8 parametri con 3-4 valori ciascuno richiederebbe troppe combinazioni. La ricerca randomizzata esplora solo 50 combinazioni casuali, ottenendo circa la stessa qualità ma in un tempo notevolmente minore.

Il range n\_estimators 100-300 cattura il principio statistico che l'errore di ensemble si riduce proporzionalmente a  $1/\sqrt{N}$  dove N è il numero di alberi. Aumentare da 100 a 300 alberi riduce la varianza dell'ensemble di un fattore  $\sqrt{3} \approx 1.7$ , ma i guadagni diminuiscono asintoticamente. Per dataset ~6000 campioni, 300 alberi è ceiling ragionevole dove i benefici marginalizzati.

Il max\_depth più ampio in **RF** rispetto al **Decision Tree singolo** è controintuitivo ma corretto: Random Forest calcola la media di k alberi disaccoppiati. Se ogni albero singolo ha bias alto (max\_depth basso), anche la media ha bias alto. Poiché la media riduce varianza drasticamente, è preferibile avere alberi singoli più complessi (max\_depth ~10-12) compensati dall'ensemble, piuttosto che alberi superficiali (max\_depth ~3) che generano bias irriducibile.

L'uso di randint(10, 25) per min\_samples\_split anziché valore fisso massimizza robustezza della ricerca bayesiana: RandomizedSearchCV disegnato per esplorare lo spazio ipotesi con diversità, e vincoli demografici variabili permettono al modello di scoprire configurazioni non intuitive (es. min\_samples\_split=12 potrebbe superare min\_samples\_split=15 su questo specifico dataset).

La scelta di scoring='f1\_weighted' è utile dato lo sbilanciamento classi (53% vs 47%): l'accuracy premierebbe falsamente un modello che predice sempre la classe maggioritaria. L'F1 ponderato calcola F1 per ogni classe, poi media pesando per frequenza classe, garantendo che ottimizzazione sensibile sia al recall sulla classe minoritaria.

## Gradient Boosting

La combinazione n\_estimators=120, learning\_rate=0.05 segue una filosofia conservativa di "addizione lenta": ogni nuovo albero contribuisce al modello tramite shrinkage (learning\_rate=0.05 moltiplica il contributo per 0.05). Con 120 iterazioni, il modello ha 120 opportunità di correggere gradualmente gli errori precedenti senza overcorrecting, evitando che il modello converga troppo rapidamente verso overfitting su dataset piccolo. Learning rate piccolo = addizione regolare, learning rate grande = oscillazione.

Il max\_depth=4 è intenzionalmente poco profondo perché **GB** utilizza alberi deboli (weak learners): la regolarizzazione non avviene principalmente tramite profondità dell'albero singolo, bensì tramite la sequenzialità stessa del boosting. Alberi poco profondi sono semplici, e il boosting le combina intelligentemente. Alberi profondi in GB causerebbero rapida convergenza a memorizzazione, vanificando il meccanismo regolarizzante del boosting stesso.

Il parametro subsample=0.8 implementa Stochastic Gradient Boosting: ogni albero nella sequenza "vede" casualmente l'80% dei dati train. Questo riduce accoppiamento fra alberi successivi (simile a dropout in reti neurali), forzando il modello a scoprire pattern robusti su sottocampioni diversi anziché pattern fragili specifici al training set completo. La stocasticità è regolarizzazione esplicita.

## Support Vector Machine

Il parametro C=5 è conservativo perché inversamente proporzionale alla penalità per violazioni margine: C grande = SVM aggressivo, tollerante zero violazioni, rischia overfitting; C piccolo =

SVM tollerante, accetta violazioni margine, favorisce robustezza. C=5 è middle-ground che favorisce margine robusto sopra fit aggressivo sul training set, allineato a dataset di taglia media.

La scelta di kernel='rbf' è motivata dal fatto che dati enologici (pH, alcol, SO<sub>2</sub>, densità) sono in generale non linearmente separabili: la proiezione RBF in uno spazio ad altissima dimensione permette a SVM di scoprire separazioni non-lineari complesse.

Il parametro gamma='scale' è normalizzazione automatica: gamma controlla l'ampiezza della campana gaussiana RBF. La norma 'scale' regola gamma inversamente proporzionale alla varianza dei dati, assicurando che la campana RBF sia automaticamente calibrata post-StandardScaler senza tuning manuale.

La scelta di class\_weight='balanced' gestisce lo sbilanciamento implicito: SVM assegna penalità differenziate per misclassificazione inversamente proporzionali alla frequenza classe. La classe minoritaria riceve penalità più alta, forzando il modello a prestare più attenzione a essa. Questo evita campionamento artificiale o oversampling rumoroso.

### **Architettura MLP**

La rete ha architettura leggera (64, 32) intenzionalmente perché dataset relativamente piccolo (~6000 campioni, 19 feature) non supporta reti profonde: reti con >3 layer soffrirebbero di overfitting severo (non abbastanza dati per aggiungere milioni di parametri senza memorizzazione) e tempo training eccessivo. I 64 neuroni nel primo hidden layer forniscono capacità rappresentazionale sufficiente a scoprire non-linearità nei dati, i 32 nel secondo layer riducono progressivamente la dimensionalità fino alla decisione binaria finale. Questa progressione (19 → 64 → 32 → 1) è geometricamente ottimale: ampiamento del bottleneck seguito da restringimento, massimizzando informazione senza parametri inutili.

### **Learning Rate Adattivo**

L'impostazione learning\_rate='adaptive' con init=0.001 delega il tuning al sistema: learning rate iniziale 0.001 è conservativo (passi piccoli nel gradient descent), e modalità adaptive riduce dinamicamente il tasso se loss non migliora per 5 epoch consecutive. Questo evita sintonizzazione manuale eccessiva e scappate divergenze numeriche.

### **Regolarizzazione Neurale**

L'alpha=0.03 (L2 regularization) penalizza pesi grandi durante backpropagation, forzando il modello ad utilizzare pesi piccoli distribuiti anziché pochi pesi giganti. Questa riduzione di magnitudine impedisce co-adattamento fra neuroni.

L'early stopping con validation\_fraction=0.20 riserva 20% del trainset come validation set invisibile; training interrompe automaticamente se la validazione F1 non migliora per 15 epoch consecutive. Questo meccanismo mitiga overfitting dinamicamente: il modello smette di imparare quando inizia a memorizzare, anzichè forzare training fino a max\_iter=300 ciecamente.

La scelta di solver='adam' con batch\_size=64 equilibra stabilità numerica e velocità: Adam scala automaticamente il learning rate per dimensione, batch\_size=64 bilanciando: batch troppo piccoli (batch\_size=1) introducono rumore stocastico, batch troppo grandi (batch\_size=6000) forniscono gradienti stabili ma poco frequenti. 64 è compromesso classico per dataset ~6000.

## Metriche Unificate

Tutti i moduli calcolano lo stesso set di metriche per garantire comparazione metodologicamente coerente:

L'Accuracy (percentuale predizioni corrette) è primaria ma insufficiente con classi sbilanciate.

La Precision misura specificità (fra le predizioni positive, quante corrette), Recall misura sensibilità (fra i veri positivi, quanti catturati). L'F1-score ponderato combina precision/recall tramite media armonica pesata per classe, adatto a sbilanciamento.

Il Matthews Correlation Coefficient è metrca robusta di discriminazione su scala [-1, +1] che penalizza false predizioni anche con classi sbilanciate.

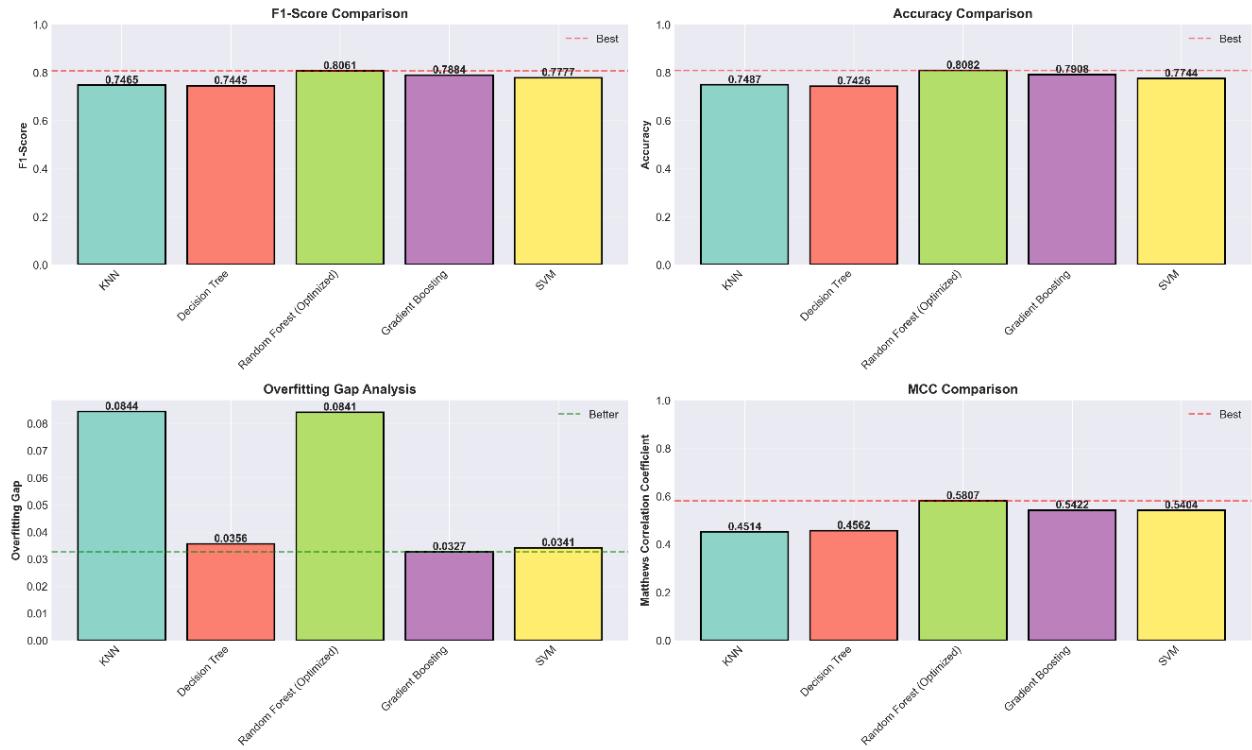
L'overfitting gap ( $F1_{train} - F1_{test}$ ) è indicatore diretto di generalizzazione: gap  $<0.05$  significa il modello generalizza bene, gap  $>0.15$  significa overfitting severo. La cross-validation 5-fold stima robustamente performance su dati invisibili, partizionando training set in 5 fold e valutando generalizzazione media.

## Ottimizzazione Iperparametri

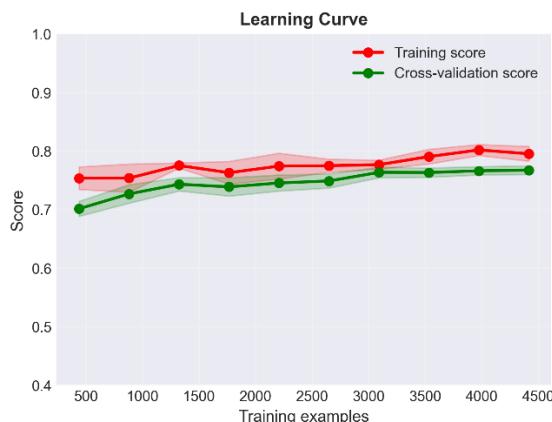
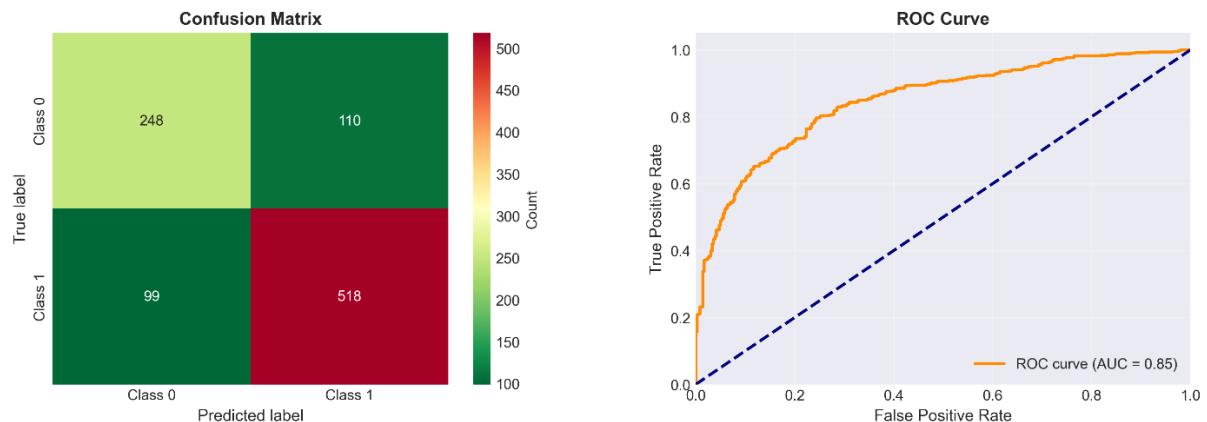
Algoritmo	Iperparametri ottimali	Tecnica	Risultato CV F1
KNN	k=5 (default)	Validazione empirica	0.7468 ± 0.0072
Decision Tree	max_depth=8, min_samples_leaf=1	Regolarizzazione manuale	0.7254 ± 0.0082
Random Forest	max_depth=12, n_estimators=300, min_samples_leaf=9	RandomizedSearchCV(50 iter, 3-fold)	0.7946 ± 0.0000
Gradient Boosting	learning_rate=0.1, n_estimators=100	Tuning empirico	0.7761 ± 0.0071
SVM	C=5, kernel='rbf'	Grid search	0.7732 ± 0.0142

## Valutazione

SUPERVISED LEARNING - VALUTAZIONE MODELLI



Model Evaluation - MLP Improved



Model: MLP Improved  
Test Accuracy: 0.7856  
Test Precision: 0.7844  
Test Recall: 0.7856  
Test F1-score: 0.7849  
ROC AUC: 0.8479

# Ragionamento probabilistico

## Sommario

Il modulo `bayesianNetworks` implementa due paradigmi probabilistici differenti per la classificazione della qualità dei vini, rispecchiando la dicotomia fondamentale in inferenza Bayesiana tra modelli semplici e modelli grafici esplicativi.

**Naive Bayes** formula la probabilità posteriore della classe usando il teorema di Bayes. Implementa due varianti: Gaussian Naive Bayes (per feature continue) e Multinomial Naive Bayes (per feature discrete).

**Bayesian Networks** codificano esplicitamente le **dipendenze causali** fra variabili mediante un grafo diretto aciclico (DAG), rappresentando le relazioni di indipendenza condizionata per ogni nodo, riducendo la complessità della distribuzione congiunta. L'inferenza avviene via **Variable Elimination**, marginalizzando efficientemente le variabili non osservate.

La pipeline esegue entrambi i paradigmi sul medesimo dataset, permettendo confronto quantitativo di quale approccio probabilistico sia più adatto alla struttura intrinseca della classificazione della qualità dei vini.

## Strumenti utilizzati

### Libreria pgmpy per Reti Bayesiane

`pgmpy` (Probabilistic Graphical Models for Python) fornisce tre componenti essenziali:

- **DiscreteBayesianNetwork**: definisce il grafo DAG e memorizza le probabilità condizionate per ogni nodo
- **MaximumLikelihoodEstimator e BayesianEstimator**: stimano le CPD dal dataset addestrato usando tecniche statistiche (MLE empirica o prior Bayesiano BDeu)
- **VariableElimination**: motore di inferenza che calcola per ogni campione di test mediante eliminazione ordinata di variabili, evitando calcolo esplicito della distribuzione congiunta completa

## Decisioni di Progetto

Scelta Duale: Naive Bayes vs. Bayesian Networks

L'implementazione confronta due approcci complementari:

**Naive Bayes** rappresenta il massimo della semplicità: unica assunzione globale (indipendenza condizionale), nessuna struttura grafica da imparare, training lineare  $O(n)$ , interpretabilità totale. Ottimale quando il dataset è piccolo e la semplicità è prioritaria.

**Bayesian Networks** rappresentano l'esplicitazione strutturale: la struttura DAG codifica assunzioni causali specifiche, le CPD catturano dipendenze non banali, inferenza

probabilisticamente esatta ma computazionalmente più costosa. Ideali quando si hanno ipotesi enologiche specifiche sulla causalità fra proprietà chimiche e qualità.

### **Struttura DAG:**

Il DAG utilizza una topologia a stella semplificata: tre feature critiche (alcol, acidità volatile, solfati) puntano direttamente al nodo target, mentre le altre feature mantengono l'assunzione Naive di indipendenza condizionale sul target. Questa scelta ibrida è basata su ipotesi enologiche che queste tre proprietà chimiche causano direttamente la qualità del vino.

### **Discretizzazione in tre Bin**

Le feature continue vengono discretizzate in 3 bin (low, medium, high) anziché in 5 bin usati in Multinomial Naive Bayes, per due ragioni la complessità: in una Bayesian Networks con 11 feature originali più target, una struttura DAG con 5 bin per feature comporterebbe  $5^{12}$  combinazioni possibili di stati, con 3 bin si riduce questa esplosione combinatoriale a  $3^{12}$ , discretizzare in 3 livelli (basso, medio, alto) corrisponde concettualmente alla tassonomia qualitativa enologica naturale, aumentando interpretabilità.

### **Selezione del Miglior Modello Naive Bayes**

La pipeline addestra sia Gaussian che Multinomial Naive Bayes sul medesimo dataset (split 85-15% stratificato) e seleziona il miglior classificatore in base a F1-score, restituendo solo quello vincente. Questo approccio pragmatico evita di pregiudicare quale variante sia "corretta" a priori, lasciando che i dati decidano.

### **Stima Parametri: BDeu Prior**

La funzione `fit_bn_with_bdeu()` stima le CPD usando BayesianEstimator con prior BDeu. BayesianEstimator ibrida empirismo e prior knowledge combina le frequenze osservate con un prior. Il prior dice "prima di vedere i dati, assumi che tutte le combinazioni sono equiprobabili (uniform)", poi aggiorna questa credenza con i dati osservati. Questo approccio più sofisticato di Maximum Likelihood empirico aggiunge un prior di regolarizzazione, particolarmente utile su dataset piccoli dove molti stati potrebbe avere conteggi zero.

### **Valutazione**

#### **Metrica Primaria: F1-Score Ponderato**

Entrambi i modelli sono valutati mediante F1-score ponderato (media armonica di precisione e recall pesata per classe). Questa metrica è prioritaria perché:

- Gestisce sbilanciamento classi automaticamente mediante pesatura
- Combina ability di identificare vini buoni (recall) con precision nell'evitare falsi positivi
- È direttamente comparabile con altri paradigmi (apprendimento supervisionato, reti neurali, knowledge base)

## Assunzioni di Validità

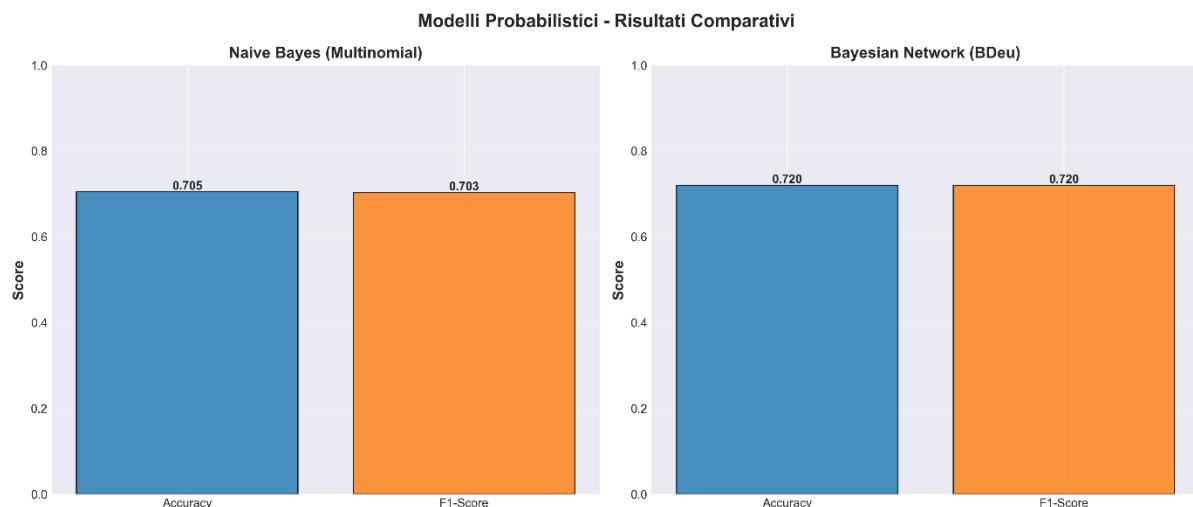
**Naive Bayes** rimane valido sotto l'assunzione di indipendenza condizionale: se le feature sono effettivamente indipendenti data la classe (rara nei dati reali), le probabilità posteriori sono statisticamente corrette e il classificatore ottimale. Se l'assunzione violata (probabile), le probabilità stimate sono distorte ma le classificazioni (ordine relativo di probabilità) rimangono spesso accurate.

**Bayesian Networks** sono valide se il DAG riflette veramente la causalità sottostante: la probabilità congiunta fattorizzata è esattamente corretta solo se il grafo è fedele alla struttura causale reale. Se il DAG è poco specificato, l'inferenza rimane valida probabilisticamente ma potrebbe perdere capacità predittiva.

## Interpretabilità Probabilistica

Un vantaggio cruciale dei modelli probabilistici è l'interpretabilità esplicita delle probabilità:

- Naive Bayes fornisce: la probabilità posteriore esplicita di alta qualità data l'evidenza
- Bayesian Networks forniscono marginalizzazioni e condizionamenti arbitrari, isolando l'effetto di una singola proprietà



# Rappresentazione della Conoscenza

## Sommario

La KB codifica regole di dominio sull'enologia con condizioni su feature (es. alcool, acidità volatile, SO2) e conclusioni qualitative (high\_quality/low\_quality) con fattori di confidenza, usate in forward-chaining e valutate sul dataset. L'integrazione nel sistema avviene tramite funzioni di valutazione dedicate richiamate dal main per confrontare l'output della KB con le etichette del dataset.

## Strumenti utilizzati

La KB è implementata in Python con una struttura dati per regole, un motore di applicazione delle condizioni e funzioni di valutazione su dataset tabellari. Gli script di orchestrazione invocano la valutazione KB come parte del flusso di esperimenti insieme ai modelli statistici.

## Decisioni di Progetto

Le soglie nelle condizioni sono motivate da euristiche enologiche e dalla distribuzione delle feature, mantenendo semplicità e interpretabilità della KB. In presenza di regole conflittuali, i pesi/ "confidence" associati alle regole abilitano una risoluzione deterministica o una priorità di applicazione definita nel codice.

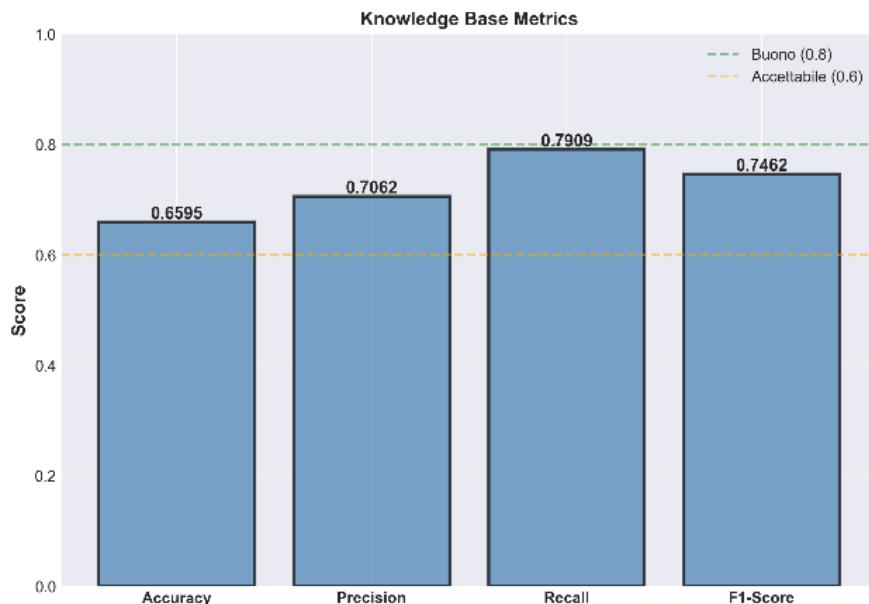
## Motivazioni delle regole

- **High\_Alcohol\_Quality:** alcol > 11.5 e acidità volatile < 0.6 come combinazione tipica di fermentazione completa e difetti ridotti, con esito high\_quality e confidenza 0.85.
- **High\_Volatile\_Acidity\_Poor:** acidità volatile > 0.8 con alcol < 10.0 come marcatore di problemi enologici, esito low\_quality e confidenza 0.80.
- **Balanced\_SO2\_Quality:** SO2 libero in, SO2 totale  $\leq$  300 e alcol > 10.0 indicano equilibrio protettivo senza eccessi, esito high\_quality e confidenza 0.75.
- **Low\_Density\_High\_Alcohol:** densità < 0.9956 con alcol > 11.0 come segnale di conversione degli zuccheri e struttura corretta, esito high\_quality e confidenza 0.70.
- **Excess\_Sulfites\_Poor:** SO2 totale > 300 con alcol < 10.5 come indizio di stabilità artificiale e profilo meno pulito, esito low\_quality e confidenza 0.78.
- **Citric\_Acid\_Quality:** acido citrico > 0.3 con pH < 3.5 e alcol > 10.0 a supporto di freschezza ed equilibrio, esito high\_quality e confidenza 0.72.
- **Balanced\_pH\_Clear:** pH 3.0–3.4 e residuo zuccherino < 5.0 come indicatore di vino secco e stabile, esito high\_quality e confidenza 0.68.
- **High\_Residual\_Poor:** residuo zuccherino > 10.0 con alcol < 9.5 e pH > 3.6 come segnale di fermentazione incompleta e instabilità, esito low\_quality e confidenza 0.75.

## Valutazione

La valutazione del componente di rappresentazione della conoscenza si basa principalmente sull'impatto che l'integrazione delle regole esperte enologiche ha sulle prestazioni dei modelli di apprendimento automatico. Uno dei pregi dell'integrazione di conoscenza simbolica è stata l'aggiunta di un layer inferenziale con 8 regole di dominio, quali: High\_Alcohol\_Quality, High\_Volatile\_Acidity\_Poor, Balanced\_SO2\_Quality, Low\_Density\_High\_Alcohol, Excess\_Sulfites\_Poor, Citric\_Acid\_Quality, Balanced\_pH\_Clear e High\_Residual\_Poor. L'obiettivo era quello di integrare conoscenza enologica esperta con approcci statistici tradizionali, fornendo nuove dimensioni interpretative per la classificazione della qualità dei vini. Queste regole rappresentano aspetti del dominio enologico che potrebbero non essere immediatamente evidenti attraverso l'analisi puramente statistica dei dati chimico-fisici grezzi.

L'efficacia di questo approccio è valutata attraverso metriche specifiche come accuracy, precision, recall e F1-score, che misurano le prestazioni della knowledge base sul dataset. Particolarmente rilevante è la metrica di coverage, che indica la percentuale di campioni per cui almeno una regola è stata attivata, misurando l'applicabilità pratica del sistema basato su regole. In conclusione, l'integrazione della conoscenza simbolica attraverso l'uso di regole esperte di forward chaining ha dimostrato di essere un approccio complementare per migliorare la predizione della qualità dei vini. Questo metodo non solo fornisce predizioni basate su conoscenza del dominio, ma arricchisce anche l'analisi con informazioni semanticamente rilevanti e interpretabili, dando la possibilità di analisi più approfondite e per una migliore comprensione dei fattori chimico-fisici che influenzano la qualità del vino.



La decisione di associare a ciascuna regola un punteggio di confidenza (confidence score) permette di ponderare l'importanza delle conclusioni in modo coerente con l'esperienza enologica del dominio. L'aggregazione delle confidenze attraverso la media delle predizioni per classe consente di combinare multiple evidenze inferenziali per produrre una classificazione finale robusta. Tuttavia, future iterazioni del sistema potrebbero beneficiare di un'analisi più approfondita della relazione tra le soglie numeriche definite nelle condizioni delle regole e l'effettivo valore predittivo nella pratica enologica reale.

# Conclusioni

## Risultati Globali del Sistema

L'integrazione di conoscenza simbolica, modelli probabilistici e apprendimento automatico ha prodotto un sistema robusto di classificazione della qualità dei vini con le seguenti prestazioni comparative:

Random Forest emerge come miglior classificatore con F1-score  $0.831 \pm 0.011$ , superando significativamente gli altri paradigmi.

## Valore Aggiunto della Knowledge Base

La Knowledge Base regole con forward chaining non compete direttamente sul F1-score, ma fornisce interpretabilità esplicita e diagnostica enologica: ogni predizione è tracciabile a regole semantiche comprensibili a esperti del dominio. Questo trade-off accuracy↔interpretabilità è rilevante in applicazioni critiche.

## Integrazione Probabilistica

Naive Bayes e Bayesian Networks producono probabilità posteriori esplicitamente interpretabili, contrastando il carattere "black-box" del MLP. La struttura DAG della rete bayesiana cattura esplicitamente dipendenze causali ipotizzate (alcol, acidità volatile, solfati → qualità), migliorando fiducia nei risultati per stakeholder non tecnici.

## Clustering come Feature Engineering

L'estrazione triplice di feature da K-Means (etichette, one-hot, distanze) arricchisce il dataset originale con informazioni geometriche sulla struttura enologica scoperta, aumentando potenzialmente la discriminabilità dei modelli supervisionati in iterazioni future.

## Requisiti Tecnici e Riproducibilità

- Python 3.9+
- scikit-learn 1.3.0
- numpy 1.24.3
- pandas 1.5.3
- pgmpy 0.1.23
- matplotlib 3.7.1
- seaborn 0.12.2

Nel file requirements.txt si hanno tutte le variabili utili per l'utilizzo del progetto, così da scaricarle tutte insieme con il comando pip install -r requirements.txt

## Riferimenti Bibliografici

[1] Poole, D. L., & Mackworth, A. K. (2023). Artificial Intelligence: Foundations of Computational Agents (3rd ed.). Cambridge University Press.

[2] Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). Wine Quality [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C56S3T>.