UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO | DIPARTIMENTO DI INFORMATICA

WineSelection

Progetto di Ingegneria della Conoscenza
A.A. 2022/2023

GRUPPO DI LAVORO

- Michele Fraccalvieri, 724204, m.fraccalvieri8@studenti.uniba.it
- Marco Angelo Lillo, 717683, m.lillo21@studenti.uniba.it

REPOSITORY

https://github.com/MarcoLillx/WineSelection

Sommario

1 Introduzione

WineSelection è un progetto realizzato per tutti gli amanti del vino. Permette di scoprire nuovi vini e come questi sono composti, permettendo inoltre di trovare il vino perfetto per ogni occasione.

Il caso di studio è stato realizzato partendo dal dataset "**Wine Quality**", messo a disposizione dalla piattaforma online "Kaggle.com" e contiene vini rossi e bianchi del *Vinho Verde Portoghese*.

La certificazione del vino e la valutazione della qualità sono elementi chiave in questo contesto. La certificazione impedisce l'alterazione illegale dei vini (per salvaguardare la salute umana) e garantisce la qualità per il mercato del vino. La valutazione della qualità fa spesso parte del processo di certificazione e può essere utilizzata per migliorare la produzione del vino (identificando i fattori più influenti) e per stratificare i vini come marchi premium (utile per stabilire i prezzi). La certificazione del vino è generalmente valutata attraverso test fisico-chimici e sensoriali. I test sensoriali si basano principalmente su esperti umani. Va sottolineato che il gusto è il senso umano meno compreso; quindi, la classificazione del vino è un compito difficile. Inoltre, le relazioni tra l'analisi fisico-chimica e sensoriale sono complesse e ancora non completamente comprese.

L'obiettivo principale del progetto è quindi quello di prevedere la qualità di un vino e di consigliare quello che più si addice alle preferenze dell'utente.

1.1 Elenco argomenti di interesse

- **Apprendimento supervisionato**: Naive Bayes, Random Forest, KNN (K-Nearest Neighbor), SVM (Support Vector Machine).
- **Sistema Esperto:** realizzazione di una base di conoscenza per la predizione della qualità di un vino.
- **Apprendimento Induttivo**: utilizzo dell'ILP per apprendere da esempi esistenti di vini di alta e bassa qualità al fine di generare regole predittive.

1.2 Sistema di Classificazione

Per il nostro sistema di predizione della qualità abbiamo scelto precisi composti chimici che, in un modo o nell'altro, influenzano la qualità del vino. Il dataset preso in esame contiene diversi dati che,

nel processo di predizione e classificazione, non hanno un impatto significativo sulla qualità; perciò, abbiamo deciso di effettuare un processo di pulizia e preprocessing dei dati prima dell'effettivo utilizzo di tale dataset.

1.3 Strumenti utilizzati

Questa parte di progetto è stata realizzata interamente in **Jupyter Notebook**.

Le librerie utilizzate nella progettazione del classificatore sono le seguenti:

- **Pandas**: per la gestione e la manipolazione del dataset
- **Numpy**, **Seaborn**: per la rappresentazione dei dati tramite la costruzione di grafici
- **Scikit-learn**: per tutte le funzioni di calcolo riguardanti i classificatori
- **Matplotlib**: per la realizzazione di alcuni grafici
- **Imblearn**: per l'oversampling dei dati

1.4 Dataset

Abbiamo deciso di utilizzare il dataset **Wine Quality** messo a disposizione dal sito "**Kaggle**". Questo dataset contiene i valori dei composti chimici del Vinho Verde Portoghese e un valore di qualità.

Il dataset contiene 6463 vini e 13 feature, di cui una sola categorica. Le feature prese in considerazione sono le seguenti:

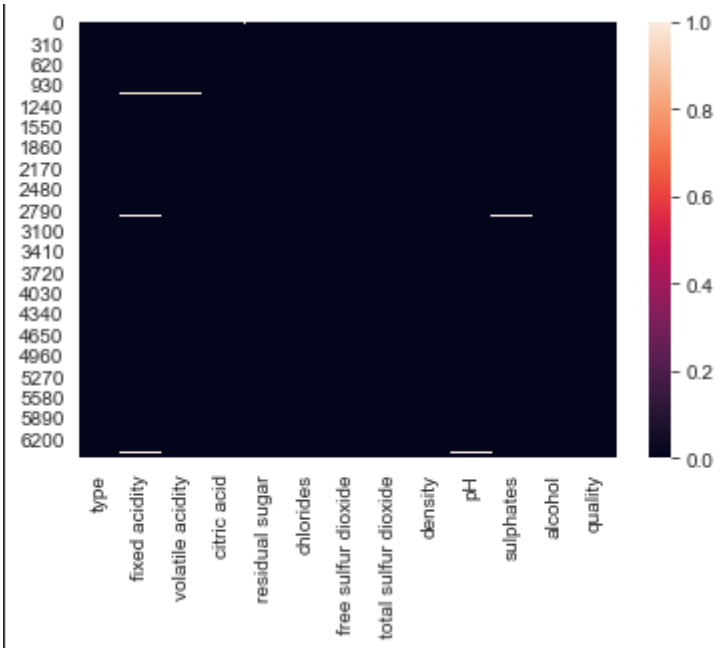
Fixed acidity	Questo tipo di acido contribuisce all'equilibrio del gusto del vino e conferisce freschezza al sapore.
Volatile acidity	Questo tipo di acido può essere percepito attraverso l'olfatto, conferisce al vino un sapore acidulo
Citric acid	Può aggiungere "freschezza" e sapore ai vini
Residual sugar	Lo zucchero dell'uva che non è stato fermentato in alcool
Chlorides	La quantità di sale nel vino
Free sulfur dioxide	Previene la crescita microbica e l'ossidazione del vino grazie alle sue proprietà antiossidanti e antimicrobiche.
Total sulfur dioxide	Viene aggiunto principalmente per uccidere i batteri nocivi e preservare la qualità e la freschezza
Density	La densità del vino può essere inferiore o superiore a quella dell'acqua. Il suo valore è determinato principalmente dalla concentrazione di alcol e zucchero.
Sulphates	I solfati sono un risultato naturale della fermentazione da parte del lievito dello zucchero nel vino in alcool.
pH	pH è una misura dell'acidità del vino. Tutti i vini idealmente hanno un livello di pH compreso tra 2,9 e 4,2
Alcohol	La percentuale alcolica contenuta nel vino. Questo parametro varia da 4,5 a 22 a seconda della categoria.
Type	Il tipo del vino (in questo caso bianco o rosso)
Quality	Il punteggio assegnato dagli esperti che va da 1 a 10

2 Analisi esplorativa e preprocessing dei dati

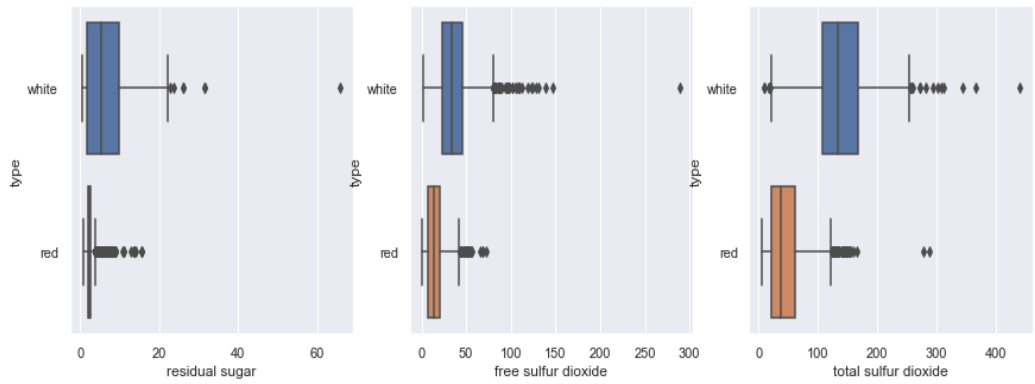
Il dataset da noi scelto presenta diversi valori non utilizzabili, che rendono i calcoli imprecisi e non veritieri. Abbiamo quindi effettuato una pulizia dei dati in modo da ridurre la dimensione del dataset e rendere più efficace il suo utilizzo.

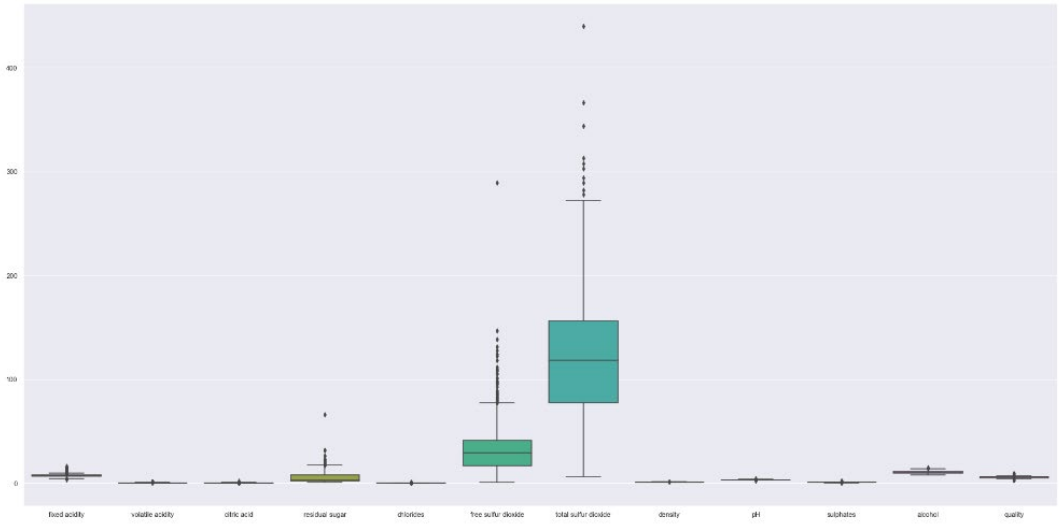
2.1 Valori nulli, duplicati e anomali

Come prima cosa, rimpiazziamo i valori nulli con la media e la mediana dei rispettivi valori. Questo perché la semplice eliminazione delle righe ridurrebbe considerevolmente la dimensione del set di dati, ma potrebbe ridurre le prestazioni dei modelli di apprendimento.



Per quanto riguarda i valori anomali, abbiamo utilizzato la formula della deviazione standard sulle feature residual sugar, free sulfur dioxide, total sulfur dioxide poiché solo queste presentavano anomalie.



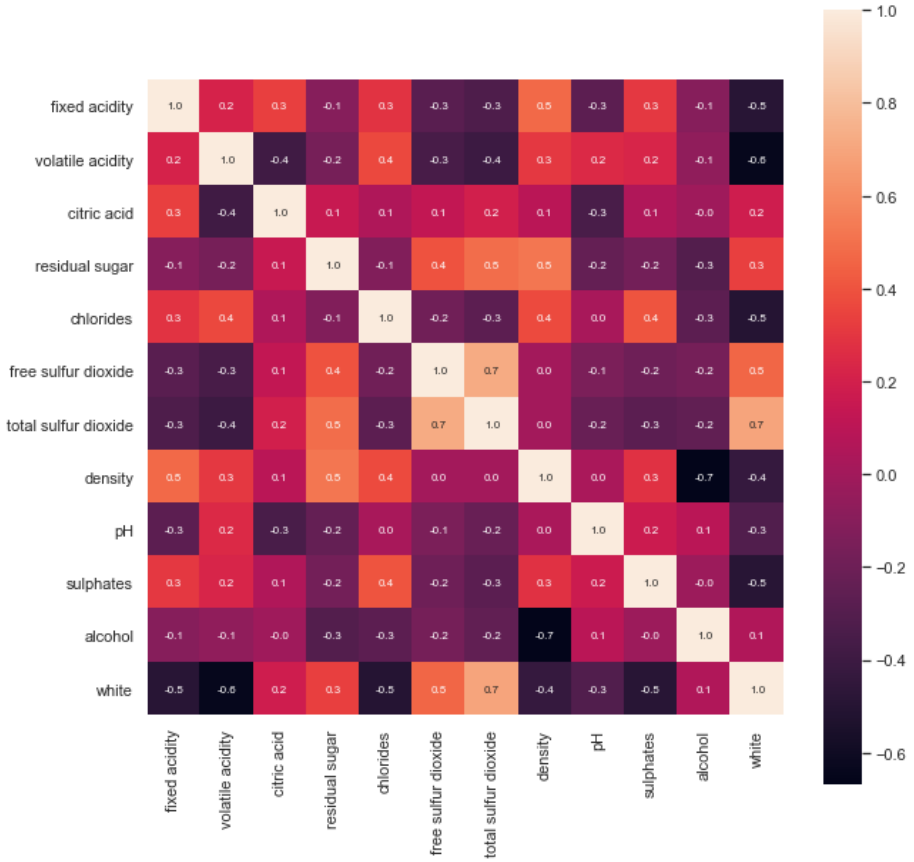


Abbiamo successivamente eliminato i valori duplicati, alleggerendo il nostro dataset di circa 1200 tuple.

2.2 Normalizzazione

La colonna 'type', essendo feature categorica, deve essere trasformata in feature numerica attraverso il "**1-hot encoding**" e quindi, trasformata in una colonna a rappresentazione binaria. Usiamo la funzione **get_dummies()** di Pandas per rimuovere la prima feature e trasformarla essenzialmente in una colonna di 1 e 0, dove 1 denota vino bianco e 0 denota vino rosso.

In seguito a queste modifiche possiamo calcolare la correlazione fra le varie features.



Data la correlazione negativa di alcohol con density, abbiamo deciso di rimuovere quest'ultima feature, in modo da rendere i risultati più attendibili.

2.3 Quality label

A questo punto la feature "quality" presenta i voti dei vini assegnati dagli esperti in una scala da 3 a 9. In questo modo però un algoritmo di predizione non risulterebbe abbastanza preciso ed efficace, soprattutto per un dataset di queste dimensioni. Abbiamo quindi scelto di classificare i voti in due variabili categoriche:

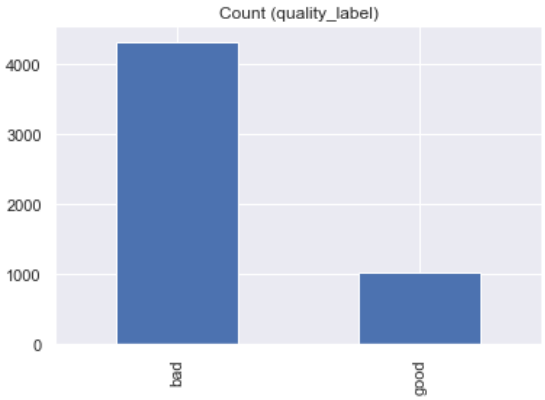
- **bad**: se il vino ha un punteggio di 6 o minore
- **good**: se il vino ha un punteggio di 7 o maggiore

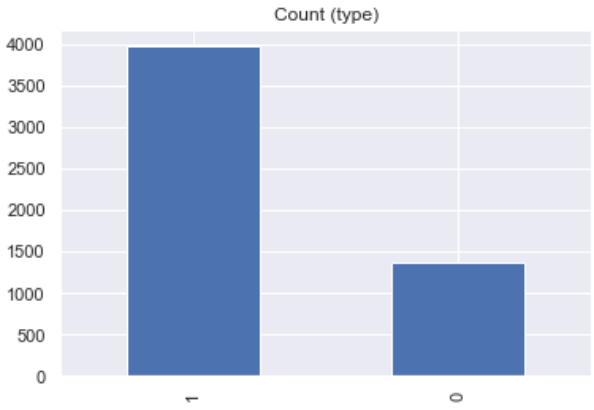
In questo modo la nostra feature target presenterà esclusivamente questi due valori e gli algoritmi di classificazione e predizione risulteranno più precisi.

Successivamente, trasformiamo questa feature in numerica tramite **LabelEncoder** e otteniamo valore 0 se bad, 1 se good. Ora tutte le nostre feature hanno valori numerici e la classificazione può procedere.

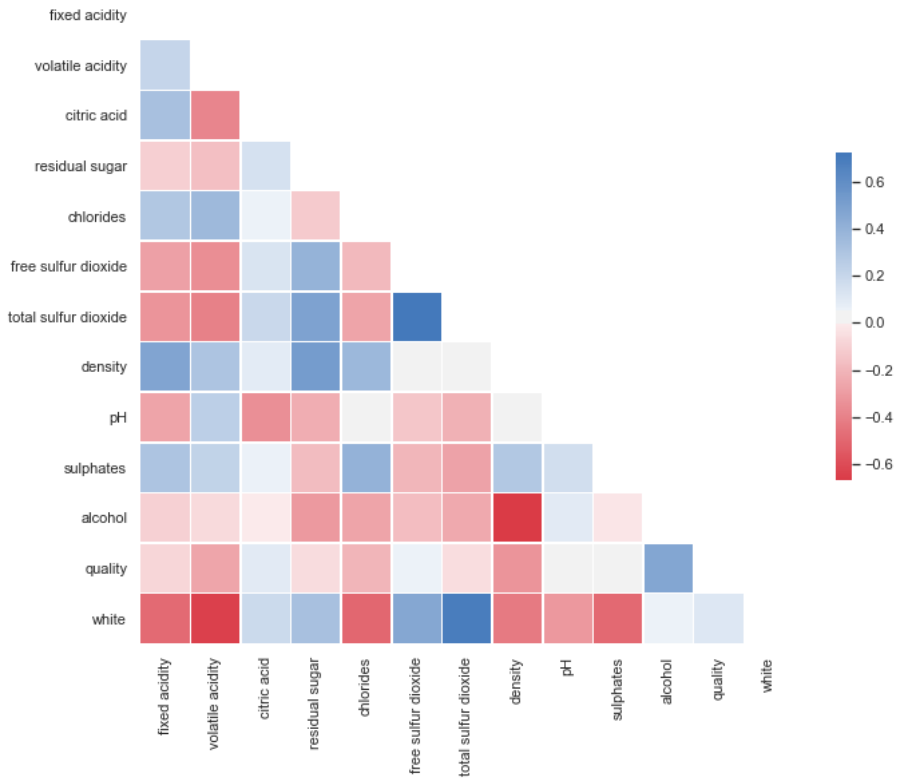
2.4 Osservazioni

Analizzando meglio le feature quality e type possiamo notare come il nostro dataset sia molto **sbilanciato**. Per quanto riguarda quality, abbiamo circa 4000 vini bad e 1000 vini good; mentre per type ci sono quasi 4000 vini bianchi e più di 1000 vini rossi.





Inoltre, da questa heatmap si evince che total sulfur dioxide e free sulfur dioxide sono correlati maggiormente con i vini bianchi e viceversa.



3. Apprendimento Supervisionato

L'apprendimento supervisionato è un tipo di approccio all'apprendimento automatico (machine learning) in cui un algoritmo viene addestrato per fare previsioni o classificazioni basate su un insieme di dati di addestramento etichettati. In altre parole, si forniscono all'algoritmo esempi di input e le corrispondenti etichette o output desiderati, e l'algoritmo impara a fare previsioni o classificazioni in base a tali esempi.

3.1 Decisioni di Progetto

L'obiettivo principale del nostro progetto è sicuramente quello di predire la qualità di un vino. Per questo motivo l'apprendimento supervisionato avrà come feature target "quality_label", precedentemente creata tramite LabelEncoder sulla feature "quality".

Per la divisione del dataset in training test e training set abbiamo deciso di non includere le feature meno importanti per quanto riguarda la qualità del vino e feature non necessarie. Queste feature sono: 'type', 'quality' e 'density'.

Per quanto riguarda i modelli scelti per la classificazione, abbiamo deciso di utilizzare **Naive Bayes**, **Random Forest**, **KNN** e **SVM**.

3.2 Metriche di valutazione

Le metriche di valutazione da noi adottate sono:

- **Accuracy**: calcola la percentuale di predizioni corrette rispetto al numero totale di predizioni.
- **Precision**: misura la proporzione di predizioni positive corrette rispetto al numero totale di predizioni positive effettuate dal modello.
- **Recall**: misura la proporzione di predizioni positive corrette rispetto al numero totale di istanze effettivamente positive nella classe di interesse
- **F1**: è una media armonica tra precisione e recall.
- **Matthews Correlation Coefficient (MCC)**: misura la qualità generale delle predizioni di un modello di classificazione binaria.
- **Geometric Mean Score**: è una misura della capacità di un modello di classificare correttamente tutte le classi.
- **ROC Curve**: è un grafico che aiuta a capire quanto un modello sia bravo nel distinguere tra due diverse categorie. Mostra quanto il modello sia efficace nel riconoscere gli esempi veri di una categoria (vero positivo) senza sbagliare troppo nel confondere gli esempi falsi (falso positivo).

Un'altra tecnica utilizzata per valutare le prestazioni dei modelli in modo accurato e affidabile è la **Cross-Validation**. Abbiamo scelto di effettuare 3 fold, dove ogni fold viene usato come set di validation, mentre 2 sono utilizzati come set di training, il tutto ripetuto 3 volte.

Per ottimizzare le prestazioni dei modelli, abbiamo eseguito un processo di **Parameter Tuning** utilizzando la tecnica della **Grid Search**. Questo ci ha permesso di esplorare una griglia di combinazioni di parametri e identificare i valori ottimali per ottenere il miglior rendimento del modello.

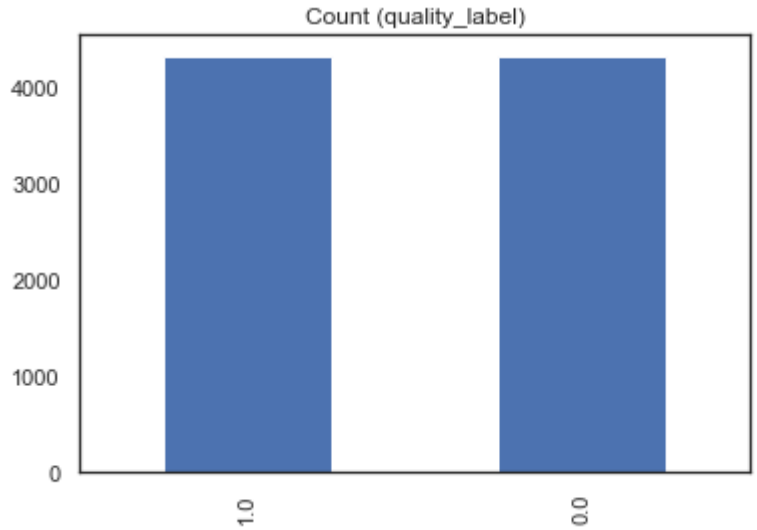
3.3 Preparazione e divisione del dataset

Come precedentemente osservato, il nostro dataset è abbastanza sbilanciato, essendo suddiviso in circa 4000 vini buoni e 1000 vini di scarsa qualità.

Per ovviare a ciò, abbiamo deciso di utilizzare **SMOTE (Synthetic Minority Oversampling Technique)**[1], una tecnica di oversampling sul set di training, utile a risolvere il problema dello sbilanciamento dei dati. Nel nostro dataset, come abbiamo visto precedentemente, ci sono più istanze di vini bianchi e meno istanze di vini rossi; quindi, i dati devono essere bilanciati per poter

[1] SMOTE - imblearn

rimuovere il "bias". Inoltre, sono stati trovati molti più vini nel range di 'quality' che va da 3 a 6, ossia ci sono molti più vini 'bad' che 'good'.



Come possiamo notare, adesso, dopo l'applicazione della tecnica SMOTE, il nostro dataset è completamente bilanciato.

Possiamo quindi effettuare la divisione del nostro dataset in **training set** e **test set**, in cui il 20% dei dati sarà utilizzato per il test set e il restante 80% per il training set.

```
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size = 0.2,random_state = 11)
```

3.4 Costruzione dei modelli

Per la costruzione di ogni modello da noi preso in esame, abbiamo deciso di creare un'istanza dello stratify k-cross validation per poter ottenere il modello migliore.

```
stratified_kfold = StratifiedKFold(n_splits=3,
                                   shuffle=True,
                                   random_state=5)
```

Dopodiché eseguiamo un processo di Parameter Tuning utilizzando la tecnica della Grid Search. Si noti che questo procedimento non è utilizzato nella costruzione del modello Naive Bayes poiché GuassianNB non accetta nessun parametro.

3.5 Naive Bayes

Il Naive Bayes[2] è un algoritmo di apprendimento supervisionato utilizzato per la classificazione e la categorizzazione di dati. Si basa sul teorema di Bayes, che è una formula statistica utilizzata per calcolare la probabilità condizionale di un evento dato un altro evento.

Abbiamo scelto questo modello per la sua semplicità e la sua efficienza computazionale, combinata alla sua capacità di trattare i dati mancanti in modo efficace.

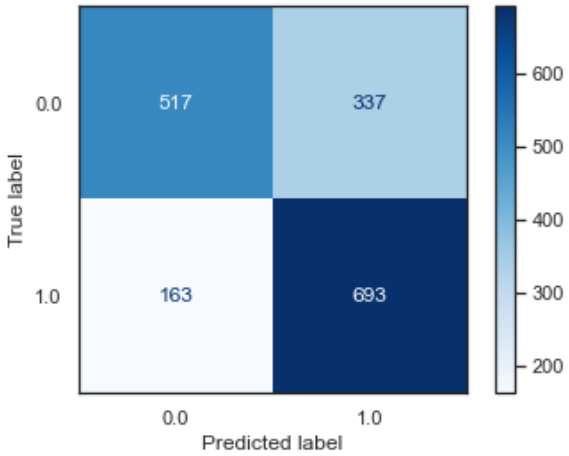
[2] Scikit-learn – Naive Bayes Classifier

A differenza degli altri modelli, il Naive Bayes non presenta iper-parametri, pertanto la griglia del GridSearchCV sarà vuota.

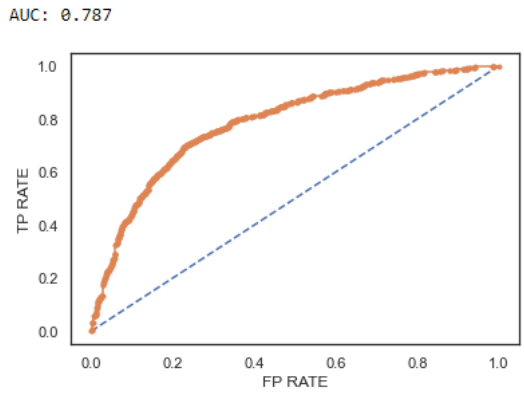
I risultati finali del test sono i seguenti:

```
accuracy: 0.7076023391812866
precision: 0.7165548258138206
recall: 0.7074829280570816
f1: 0.7044717060174399
matthews_corrcoef: 0.423940700309573
geometric_mean_score: 0.7074829280570816
```

3.5.1 Rappresentazione dei risultati



Matrice di Confusione[3]: il numero di falsi negativi è decisamente molto alto rispetto alle previsioni corrette, mentre il numero di falsi positivi è un valore discreto rispetto al numero di previsioni corrette. Il modello ha quindi scarse capacità di previsione.



ROC Curve[4]: L'AUC (Area Under the Curve) è leggermente positiva, con un valore di 0.787. Questo indica che il modello ha una discreta capacità di discriminare tra le classi positive e negative.

3.6 Random Forest

L'algoritmo Random Forest[5] è un metodo di apprendimento automatico che semplifica la complessità dell'addestramento di molti alberi decisionali per migliorare le prestazioni predittive e ridurre l'overfitting.

[3] Matrice di Confusione - Wikipedia
[4] ROC Curve - Wikipedia
[5] Scikit-learn – Random Forest Classifier

3.6.1 Parameter Tuning

Gli iper-parametri scelti (con i relativi valori) sono i seguenti:

- n_estimators: [50,100, 200]
- max_depth: [6,10, 12]
- random_state: [0,4,5,11,101]

n_estimators: Questo iperparametro controlla il numero di alberi decisionali nella foresta. Un numero maggiore di alberi tende a migliorare la capacità predittiva del modello, ma aumenta anche il tempo di addestramento.

max_depth: Questo iperparametro determina la massima profondità di ciascun albero decisionale. Una profondità maggiore può consentire al modello di adattarsi meglio ai dati di addestramento, ma può anche aumentare il rischio di overfitting.

random_state: Questo iperparametro controlla la casualità nell'addestramento degli alberi. Fissando un valore specifico per random_state, si rende l'addestramento riproducibile, il che è utile per scopi di test e validazione.

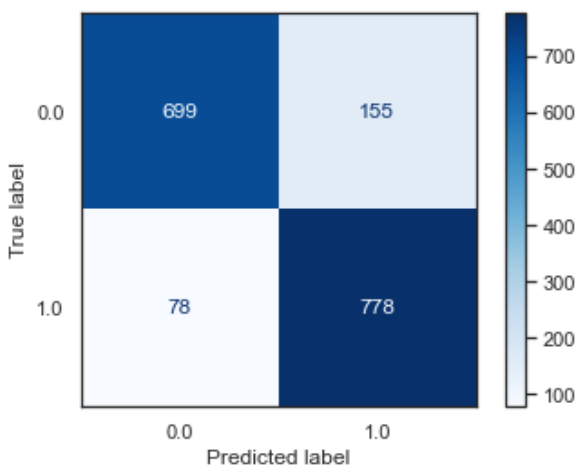
Effettuando quindi il calcolo per la ricerca del modello migliore, otteniamo i seguenti parametri:

- max_depth: 12
- n_estimators: 200
- random_state: 101

I risultati finali del test sono i seguenti:

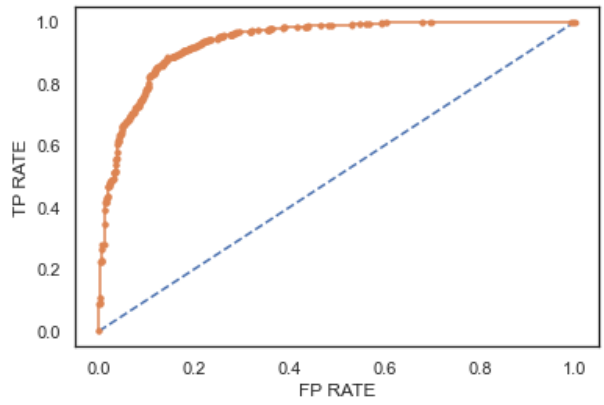
```
accuracy: 0.8637426900584795
precision: 0.8667415693139165
recall: 0.8636898378165423
f1: 0.8634512497005509
matthews_corrcoef: 0.7304250320583558
geometric_mean_score: 0.8636898378165423
```

3.6.2 Rappresentazione dei risultati



Matrice di Confusione: Possiamo notare come questo modello, rispetto al precedente, abbia una capacità sicuramente migliore di effettuare previsioni corrette per entrambe le classi. Infatti, il numero di falsi positivi (155) e falsi negativi (78) sono valori decisamente accettabili a fronte delle circa 700 previsioni corrette per classe.

AUC: 0.936



ROC Curve: L'AUC (Area Under the Curve) indica che il modello ha un'ottima capacità di discriminare tra le classi positive e negative. Il punteggio ottenuto (0.936) suggerisce una buona performance complessiva del modello.

3.7 K-Nearest Neighbor (KNN)

Il KNN[6] è un algoritmo di classificazione che mira a determinare la classe di appartenenza di un dato in input, cercando tra tutti gli esempi di addestramento, quello più vicino al dato di input in base alla metrica desiderata. Questo algoritmo è particolarmente adatto per problemi in cui i dati sono distribuiti in modo sparso o complesso.

Abbiamo scelto questo modello in quanto risulta semplice da comprendere e interpretare, in grado di adattarsi a dati complessi e non lineari. Inoltre, funziona bene con dataset di piccole dimensioni, e può risultare robusto rispetto al rumore nei dati.

3.7.1 Parameter Tuning

Gli iper-parametri scelti (con i relativi valori) sono i seguenti:

- n_neighbors: [10, 11, 12, 13]
- weights: ['distance']
- algorithm: ['ball_tree', 'kd_tree']
- leaf_size: [12, 11, 13]
- p: [1]

n_neighbors: Questo parametro rappresenta il numero di vicini (K) da considerare quando si effettua una previsione per un nuovo punto dati.

weights: Questo parametro definisce il metodo di pesatura dei vicini durante il calcolo della previsione. In particolare, 'distance' ci permette di dare più importanza ai vicini più vicini.

algorithm: Questo parametro definisce l'algoritmo utilizzato per trovare i vicini più vicini in modo efficiente.

leaf_size: Il parametro "leaf_size" è specifico per gli algoritmi "ball_tree" e "kd_tree". Indica la dimensione massima delle foglie nell'albero utilizzato per la ricerca dei vicini.

p: Questo parametro specifica il tipo di norma utilizzato per calcolare la distanza tra i punti. p=1 corrisponde alla distanza di Manhattan e p=2 alla distanza Euclidea.

[6] Scikit-learn – K-Nearest Neighbor Classifier

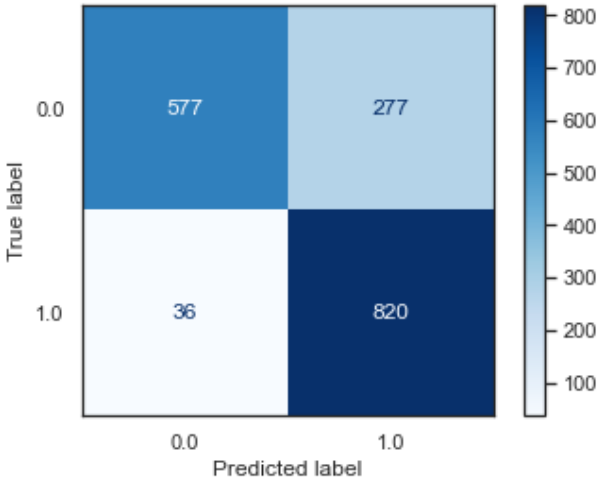
Effettuando quindi il calcolo per la ricerca del modello migliore, otteniamo i seguenti parametri:

- n_neighbors: 10
- weights: 'distance'
- algorithm: 'ball_tree'
- leaf_size: 12
- p: 1

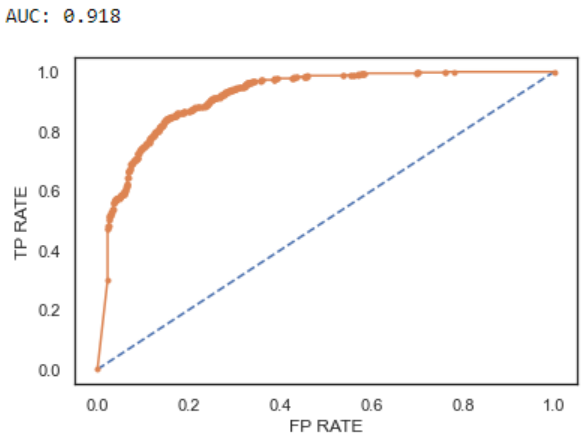
I risultati finali del test sono i seguenti:

```
accuracy: 0.8169590643274853
precision: 0.844382796920565
recall: 0.8167939766683447
f1: 0.8131865715479409
matthews_corrcoef: 0.6606009256204013
geometric_mean_score: 0.8167939766683447
```

3.7.2 Rappresentazione dei risultati



Matrice di Confusione: Possiamo notare come questo modello abbia una buona capacità di fare previsioni corrette, con un numero di falsi positivi più elevati dei falsi negativi. Ciò potrebbe suggerire che il modello è leggermente incline a effettuare previsioni corrette quando la classe reale è positiva.



ROC Curve: Il grafico indica una buona capacità del modello di discriminare tra le due classi.

3.8 Support Vector Machine

L'SVM[7] è un classificatore il cui obiettivo è trovare un iperpiano che possa separare un set di dati in due classi nel modo migliore possibile. Questo piano è definito in modo tale che massimizzi la distanza tra i punti più vicini delle diverse classi.

La scelta di questo modello è dovuta principalmente alla sua particolare adattabilità alle classificazioni binarie, in quanto appunto dividono i dati in due classi tramite un iperpiano. Inoltre, l'SVM può gestire problemi con un numero moderato di feature e relazioni non-lineari attraverso l'uso di kernel.

3.8.1 Parameter Tuning

Gli iper-parametri scelti (con i relativi valori) sono i seguenti:

- C: [0.08, 0.1, 1, 10, 100],
- gamma: [1, 0.1, 0.01, 0.001, 0.0001],
- kernel: ['rbf']

C è l'iperparametro di regolarizzazione delle SVM e controlla il compromesso tra la massimizzazione del margine tra le classi e la riduzione dell'errore di classificazione.

Gamma controlla la flessibilità del modello SVM. Valori più alti di gamma portano a modelli più complessi e possono causare l'overfitting, mentre valori più bassi di gamma producono modelli più semplici ma potrebbero sottostimare la complessità dei dati.

Il kernel specifica la funzione matematica utilizzata per trasformare i dati in uno spazio in cui le classi possono essere separabili da un iperpiano. In particolare, abbiamo scelto RBF (Radial Basis Function) perché è un kernel flessibile che può gestire sia dati lineari che dati non lineari.

Effettuando quindi il calcolo per la ricerca del modello migliore, otteniamo i seguenti parametri:

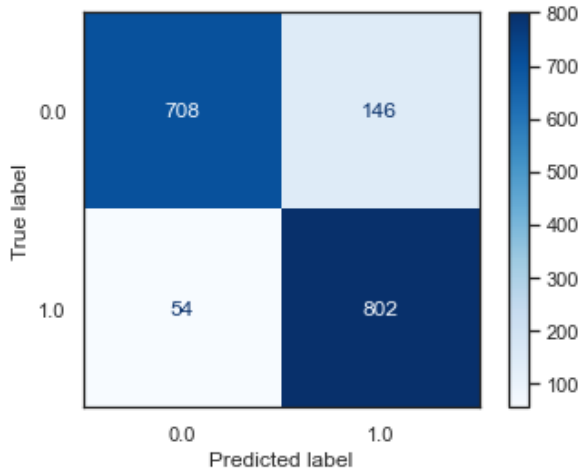
- C: 10
- gamma: 0.1
- kernel: 'rbf'

I risultati finali del test sono i seguenti:

```
accuracy: 0.8830409356725146
precision: 0.8875627097245755
recall: 0.8829778502484187
f1: 0.8826864393756448
matthews_corrcoef: 0.7705269194694495
geometric_mean_score: 0.8829778502484187
```

[7] Scikit-learn – Support Vector Machine Classifier

3.8.2 Rappresentazione dei risultati



Matrice di Confusione: Il modello presenta un'ottima capacità di predizione, seppur ci siano ancora diversi errori. Questi però sono di una misura decisamente inferiore rispetto al numero di previsioni corrette effettuate.

3.9 Conclusioni

Raggruppiamo tutti i risultati ottenuti dai vari modelli utilizzati nella seguente tabella:

Model	Train Accuracy	Test Accuracy	Precision	Recall	F1	Matthews Corrcoef	Geometric mean score
Naive Bayes	0.712	0.708	0.717	0.707	0.704	0.424	0.707
Random Forest	0.852	0.864	0.867	0.864	0.863	0.730	0.864
KNN	0.812	0.817	0.844	0.817	0.813	0.661	0.817
SVM	0.866	0.883	0.888	0.883	0.883	0.771	0.883

Come possiamo vedere dai grafici precedenti, il modello migliore è risultato il Support Vector Machine (accuracy dell'88%) seguito dal Random Forest (accuracy dell' 86%) con indice di profondità pari a 12. Nel complesso quasi tutti i modelli si sono comportati circa in egual modo, a differenza del Naive Bayes che ha restituito un Accuracy dell'70% circa.

L'accuratezza ottenuta dai vari modelli di apprendimento non è eccellente. Questo può essere spiegato dal fatto che le caratteristiche di qualità sono in realtà delle metriche selezionate da un esperto che tiene conto di ulteriori caratteristiche, che non sono quelle fisiche o chimiche, e quindi non sono presenti in questo dataset. La valutazione della qualità è una valutazione sensoriale, basata sul processo di degustazione, solo per citarne alcune avremmo:

- Altre caratteristiche fisiche rilevanti non presenti, come opacità e corposità.
- Come abbiamo visto, alcune delle caratteristiche che abbiamo possono influenzare l'odore e il sapore del vino, ma sicuramente non abbiamo tutti i dati necessari, ad esempio la varietà dell'uva o il tipo.
- Non abbiamo il tannino, una delle caratteristiche fondamentali del vino. Nel vino è la presenza di composti fenolici che conferisce amarezza e astringenza al vino.

4 Knowledge Base (KB)

Una **Knowledge Base (KB)** è un sistema o una raccolta organizzata di informazioni, dati, regole o conoscenze precedentemente acquisite e utilizzate per supportare il processo di apprendimento automatico o di ragionamento di un algoritmo o di un sistema. Per il nostro caso, abbiamo realizzato due diverse KB in linguaggio **Prolog** utilizzando il programma **Swi-Prolog**, a cui ci siamo interfacciati tramite la libreria **pyswip**, scritta per il linguaggio Python. All'interno delle Basi di Conoscenza, l'utente potrà interagire con il sistema attraverso delle query.

4.1 Wine Quality Prediction

La prima Knowledge Base descrive un modello per la predizione della qualità di un vino basandosi sui loro composti chimici. La base di conoscenza è stata interamente realizzata facendo riferimento ai dati presenti nel dataset usato per la classificazione.

4.1.1 Decisioni di progetto

In questa Knowledge Base, i fatti sono stati scritti secondo la rappresentazione tripla "**prop(Individuo, Proprietà, Valore)**" in cui l'individuo corrisponde al soggetto, la Proprietà corrisponde al verbo e il Valore corrisponde all'oggetto.

La rappresentazione tripla semplifica l'interrogazione delle conoscenze nella knowledge base, e permette di interrogare la KB sia sugli individui e sui valori, sia sulle proprietà.

Per quanto riguarda l'euristica utilizzata nel nostro algoritmo di classificazione, abbiamo scelto l'**Hill Climbing** poiché utilizza poca quantità di memoria (complessità spaziale). È un algoritmo molto rapido e veloce e consente quindi, di trovare soluzioni ottimali per determinati problemi in un tempo polinomiale, mentre in altri casi non si può garantire la convergenza verso un ottimo globale. Inoltre, seleziona soltanto la via migliore tra tutte quelle immediatamente disponibili senza considerare le conseguenze della scelta nei passi successivi.

Infine, per la realizzazione in Python del nostro programma, abbiamo deciso di utilizzare una funzione di inizializzazione che chiama due funzioni:

- **listProp()**: per la conversione dei fatti in esempi del tipo esempio(Classe, [feature1 = value, feature2 = value,…]). Questa procedura è essenziale per il funzionamento del nostro algoritmo ILP. Inoltre, la conversione verrà effettuata anche su eventuali vini aggiunti dall'utente (*Paragrafo 4.1.4*).
- **learn()**: per l'apprendimento delle regole. Questa funzione crea una lista di esempi composti dalla classe bad e una dalla classe good. Il motivo di tale scelta progettuale è descritto nel *Paragrafo 4.1.3*.

```
def initialization(wines):
    pq.listProp(wines)
    pq.learn()
```

4.1.2 Fatti e Regole

La nostra Base di Conoscenza presenta 41 vini diversi, ognuno dei quali possiede 11 proprietà con i rispettivi valori. Le proprietà presenti nella KB sono state scelte individualmente dal dataset di partenza, in base alla loro incidenza sulla qualità del vino (per questioni di privacy, il dataset non fornisce i nomi e i prezzi dei vini in questione). Queste scelte sono state frutto di ricerche approfondite sul web e studio autonomo tramite la classificazione dello stesso dataset.

Le proprietà realizzate nella nostra Base di Conoscenza sono le seguenti (accanto un esempio di utilizzo):

- fixed_acidity prop(wine_1, fixed_acidity, 8.1)
- volatile_acidity prop(wine_1, volatile_acidity, 0.27)
- citric_acid prop(wine_1, citric_acid, 0.41)
- residual_sugar prop(wine_1, residual_sugar, 1.45)
- chlorides prop(wine_1, chlorides, 0.033)
- free_sulfur_dioxide prop(wine_1, free_sulfur_dioxide, 11.0)
- total_sulfur_dioxide prop(wine_1, total_sulfur_dioxide, 63.0)
- sulphates prop(wine_1, sulphates, 0.56)
- alcohol prop(wine_1, alcohol, 12.0)
- color prop(wine_1, color, white).
- quality prop(wine_1, quality, bad).

Per quanto riguarda le regole, abbiamo realizzato nove classi che hanno come funzione quella di classificare i valori delle proprietà in low, medium o high. Questa classificazione è basata su una statistica che abbiamo generato durante la fase di apprendimento supervisionato. Di seguito vengono riportati i valori ottenuti.

	Red wine			**White wine**		
	Min	**Max**	**Mean**	**Min**	**Max**	**Mean**
fixed acidity	4.6	15.9	8.3	3.8	14.2	6.9
volatile acidity	0.1	1.6	0.5	0.1	1.1	0.3
citric acid	0.0	1.0	0.3	0.0	1.7	0.3
residual sugar	0.9	15.5	2.5	0.6	65.8	6.4
chlorides	0.01	0.61	0.08	0.01	0.35	0.05
free sulfur dioxide	1	72	14	2	289	35
total sulfur dioxide	6	289	46	9	440	138
density	0.990	1.004	0.996	0.987	1.039	0.994
pH	2.7	4.0	3.3	2.7	3.8	3.1
sulphates	0.3	2.0	0.7	0.2	1.1	0.5
alcohol	8.4	14.9	10.4	8.0	14.2	10.4

Le classi realizzate sono le seguenti (accanto un esempio di utilizzo e la proprietà a cui si riferiscono):

• fa_class	prop(F, fa_class, low):- F < 6.	fixed_acidity
• va_class	prop(VA, va_class, medium):- VA >= 0.3, VA =< 0.6.	volatile_acidity
• ca_class	prop(CA, ca_class, high):- CA >= 0.5.	citric_acid
• rs_class	prop(RS, rs_class, low):- RS < 3.	residual_sugar
• cl_class	prop(CL, cl_class, medium):- CL >= 0.04, CL < 0.08.	chlorides
• fs_class	prop(FS, fs_class, high):- FS >= 40.	free_sulfur_dioxide
• ts_class	prop(TS, ts_class, low):- TS < 50.	total_sulfur_dioxide
• s_class	prop(S, s_class, medium):- S >= 0.4, S < 0.8.	sulphates
• a_class	prop(A, a_class, high):- A >= 11	alcohol

4.1.3 Inductive Logic Programming (ILP)

L'**Inductive Logic Programming**[8] è un approccio all'apprendimento automatico che mira a estrarre regole logiche dai dati, utilizzando la logica simbolica come strumento principale di rappresentazione delle conoscenze. Tramite un algoritmo ILP abbiamo potuto indurre delle regole che servono per classificare la qualità di un vino. All'interno del nostro programma induttivo le feature le chiameremo attributi, mentre valori sarà una lista di valori associati all'attributo.

Il nostro programma è composto da due fasi principali:

- Fase di Apprendimento
- Fase di Classificazione

La fase di Apprendimento ha il compito di apprendere regole dagli esempi. La fase di Classificazione si occupa della classificazione dell'oggetto in questione.

Fase di Apprendimento:

La funzione principale di questa fase è **apprendi(Classe)** che raccoglie tutti gli esempi in una lista, e, tramite questi, elabora una descrizione per la classe composta da una lista di coppie attributo-valore e asserisce la corrispondente regola. Gli attributi sono definiti con rappresentazione binaria nome-valore, per esempio **attributo(fixAcid_Class, [low, medium, high])**.

A questo punto, viene creata una lista di congiunzioni di coppie attributo-valore che copre almeno un esempio positivo per la classe di interesse e nessuno negativo (**Conge**). Questa lista viene usata per apprendere una possibile congiunzione di coppie attributo-valore per la classe di interesse,

[8] Inductive Logic Programming - Wikipedia

tramite la funzione **apprendi_cong(Esempi, Classi, Conge)**. Questa funzione sceglie una singola congiunzione nella lista che faccia da condizione, e filtra fra gli esempi di training che soddisfano tale condizione, restituendo la lista degli esempi correttamente filtrati.

Per scegliere la condizione migliore viene utilizzata la funzione **scegli_cond(Esempi, Classe, AttVal)** che va a cercare le coppie attributo-valore che sono accettabili e li ordina in base ad un punteggio (euristica che servirà per ridurre l'elevata complessità del programma). Successivamente, avendo definito una graduatoria di possibili coppie attributo-valore, che possono essere candidate a far parte della descrizione della classe, andiamo a prendere la migliore coppia, cioè quella che ha il punteggio migliore.

L'euristica utilizzata nel nostro programma determina ogni volta la migliore coppia attributo-valore da aggiungere alla congiunzione rigettando le altre possibilità. In particolare, viene applicata una ricerca Hill Climbing, cioè un algoritmo di ricerca **Greedy**, ma che potrebbe portare all'incompletezza.

Nel nostro caso, l'euristica è contenuta nella funzione **punti(Esempi, Classe, AttVal, Punti)**, la quale cerca una coppia attributo-valore candidata a far parte della congiunzione per la descrizione di una classe.

A questo punto, vengono presi i possibili Valori chiamando la funzione **candidato(Esempi, Classe, Att = Val)**. Successivamente controlliamo se questa coppia è adatta per far parte della Congiunzione utile per caratterizzare una Classe = **adatto(AttVal, Esempi, Classe)**:

- cerchiamo almeno un esempio negativo, all'interno della lista di esempi

Infine:

- andiamo a filtrare in Esempi tutte le coppie che soddisfano Attributo-Valore, generando una nuova lista chiamata Esempi1
- verifichiamo da quanti elementi è composta Esempi1
- effettuiamo un conteggio degli oggetti di una determinata Classe all'interno di Esempi1 (deve esserci almeno un esempio positivo)
- sulla variabile Punti applichiamo la correzione di Laplace[9]

Nel nostro caso, il predicato punti(Esempi, Classe, AttVal, Punti) determina la migliore coppia Attributo-Valore nello stato attuale, rigettando le altre possibilità, e cioè prende la coppia con maggior probabilità Laplaciana di discriminare gli esempi positivi da quelli negativi. Viene applicata questa correzione poiché per insiemi di cardinalità piccola il rapporto (esempi positivi/esempi totali) viene influenzato maggiormente dal numero di esempi positivi.

Fase di Classificazione:

La funzione principale di questa fase è **classifica(Oggetto, Classe)** che crea una descrizione della classe e chiama la funzione **soddisfa(Oggetto, Congiunzione)**. Quest'ultima verifica se esiste un attributo all'interno della descrizione dell'oggetto inquadrato il cui valore sia diverso da quello presente in ciascuna coppia attributo-valore.

[9] Laplace smoothing – Wikipedia

4.1.4 Interfaccia Python

Il programma da noi realizzato permette all'utente di eseguire diverse istruzioni:

1. **Visualizzare tutte le composizioni dei vini**: Permette di visualizzare tutti vini con i corrispondenti composti chimici. Ciò avviene richiamando una regola Prolog che, specificando la proprietà "color" (il colore del vino), è in grado di restituire tutti i vini con i relativi composti chimici

```
List of all the wines:
wine_1 > {'Fixed Acidity': '8.1', 'Volatile Acidity': '0.27', 'Citric Acidity': '0.41', 'Residual
 Sugar': '1.45', 'Chlorides': '0.033', 'Free Sulfur Dioxide': '11.0', 'Total Sulfur Dioxide': '63.0',
  'Sulphates': '0.56', 'Alcohol': '12.0', 'Color': 'white', 'Quality': 'bad'}
wine_2 > {'Fixed Acidity': '8.6', 'Volatile Acidity': '0.23', 'Citric Acidity': '0.4', 'Residual
 Sugar': '4.2', 'Chlorides': '0.035', 'Free Sulfur Dioxide': '17.0', 'Total Sulfur Dioxide': '109.0',
  'Sulphates': '0.53', 'Alcohol': '9.7', 'Color': 'white', 'Quality': 'bad'}
```

2. **Aggiungere un nuovo vino**: Permette di aggiungere un nuovo vino al dataset e, in caso non si conosca la qualità del vino, questa verrà predetta dall'algoritmo di ILP, il quale classificherà il vino in base ai composti chimici inseriti dall'utente

```
Enter the name of the wine to add to the Knowledge Base [without spaces]:
> VinhoVerde

Enter fixed acidity [> 0]: 10.6
Enter volatile acidity [> 0]: 1.2
Enter citric acid [>= 0]: 0.40
Enter residual sugar [> 0]: 15
Enter chlorides [> 0]: 0.025
Enter free sulfur dioxide [> 0]: 19
Enter total sulfur dioxide [> 0]: 45
Enter sulphates [> 0]: 0.80

Enter alcohol [> 0]: 10
Enter color [red/white]: red

Do you know how's the quality of this wine?[y/n]:
> n
```

```
The wine entered belongs to the class: bad
vinhoverde wineid
10.6 fixed_acidity
1.2 volatile_acidity
0.40 citric_acid
15 residual_sugar
0.025 chlorides
19 free_sulfur_dioxide
45 total_sulfur_dioxide
0.80 sulphates
10 alcohol
red color
bad quality

Wine added!
```

3. **Predire la qualità di un vino tramite l'algoritmo di ILP**: L'utente potrà inserire i valori dei composti chimici di un vino e conoscerne la qualità, tramite una predizione calcolata dall'algoritmo di ILP

```
Enter fixed acidity [> 0]: 10.5
Enter volatile acidity [> 0]: 3
Enter citric acid [>= 0]: 0.32
Enter residual sugar [> 0]: 17
Enter chlorides [> 0]: 0.015
Enter free sulfur dioxide [> 0]: 42
Enter total sulfur dioxide [> 0]: 65
Enter sulphates [> 0]: 0.40
Enter alcohol [> 0]: 10
Enter color [red/white]: white

The wine entered belongs to the class: bad
```

4.2 Wine Recommender System

La seconda Knowledge Base descrive un modello per la raccomandazione di un vino basandosi sulle loro caratteristiche. La costruzione di questa KB è stata interamente realizzata tramite le informazioni apprese dal sito compravini.it.

4.2.1 Decisioni di Progetto

In questa Knowledge Base abbiamo deciso di optare per una realizzazione meno complessa della precedente, in quanto non abbiamo fatto uso di particolari algoritmi scritti in Prolog.

La scelta di aggiungere regole per la ricerca del prezzo è dovuta alla funzione **closest_value(price_list, target)**. Questo metodo crea una lista di prezzi vicini a quello dato in input per cercare il prezzo più vicino. In questo modo, anche se l'utente immette un prezzo non esistente, il programma trova il prezzo più vicino a quello ricercato.

4.2.2 Fatti e Regole

La nostra Base di Conoscenza presenta 30 vini italiani diversi, ognuno dei quali possiede 7 proprietà con i rispettivi valori. I fatti sono stati scritti nel seguente formato:

- **wine(Id, Title, Type, Province, Price, Taste, Food)**

In cui:

- Id è l'identificatore univoco del vino
- Title è il nome completo del vino (compresa l'annata)
- Type è il colore del vino
- Province è la regione di provenienza del vino
- Price è il prezzo del vino
- Taste è il sapore del vino
- Food è il cibo con cui il vino si abbina meglio

Per quanto riguarda le regole, queste si dividono in due gruppi in base alla loro funzionalità:

- Regole per la ricerca di Title: hanno lo scopo di ricercare il titolo del vino, sapendo una o più caratteristiche che lo compongono. Un esempio per la ricerca del titolo sapendo il prezzo, il sapore e il cibo è:
  - **title_price_taste_food(Title, Price, Taste, Food) :- wine(_, Title, _, _, Price, Taste, Food).**
- Regole per la ricerca di Price: hanno lo scopo di ricercare il prezzo del vino, sapendo una o più caratteristiche che lo compongono. Un esempio per la ricerca del prezzo sapendo il sapore e il cibo è:
  - **price_taste_food(Price, Taste, Food) :- wine(_,_, _,_, Price, Taste, Food).**

4.2.3 Funzionalità del programma

Il programma è suddiviso in tre diverse funzioni principali che collaborano per fornire le raccomandazioni di vino.

recommender()

Questa è la funzione principale del programma. Raccoglie le preferenze dell'utente riguardo al tipo di vino, al prezzo, al gusto e all'abbinamento con il cibo attraverso una serie di domande. Successivamente, determina se tutte le preferenze sono specificate o se alcune preferenze sono

mancanti. In base a questa valutazione, chiama la funzione appropriata per trovare i vini corrispondenti e successivamente stampa le raccomandazioni.

find_matching_wine(preferences)

Questa funzione è responsabile della ricerca di vini che corrispondono alle preferenze dell'utente. Verifica le preferenze dell'utente in ciascuna categoria (tipo, prezzo, gusto e cibo) e chiama funzioni per effettuare la ricerca basata su diverse combinazioni di preferenze. L'output di questa funzione può contenere più di un vino che corrisponde alle preferenze dell'utente.

print_matching_wine(wine)

Questa funzione stampa i dettagli del vino raccomandato, come titolo, tipo, regione, prezzo, gusto e abbinamento con il cibo. Inoltre, seleziona casualmente uno dei vini raccomandati se ce ne sono più di uno.

4.2.4 Interfaccia Python

Il programma da noi realizzato permette all'utente di eseguire diverse istruzioni:

- **Visualizzare i vini presenti nella KB**:
  - Visualizzare l'intera lista di vini presenti nel catalogo

```
 1 -- Print all wines
 2 -- Print a random wine
 3 -- Return to main menu

Enter a number: 1
>Bellavista Vittorio Moretti 2013
>Baglio di Pianetto 2007 Ficiligno White
>Quintarelli Giuseppe Amarone della Valpolicella Classico 2009
>Stemmari 2013 Dalila White
>Stemmari 2013 Nero d'Avola
>Terre di Giurfo 2011 Mascaria Barricato (Cerasuolo di Vittoria)
>Passito di Pantelleria 2021
```

  - Visualizzare un solo vino casuale presente nel catalogo

```
 1 -- Print all wines
 2 -- Print a random wine
 3 -- Return to main menu

Enter a number: 2

>Cantina Diomede Annibale Nero di Troia 2015
```

- **Ricerca dei vini**:
    - Ricerca di un vino nel catalogo tramite il suo nome

```
Enter a number: 2

 1 -- Find by name
 2 -- Find by ID
 3 -- Return to Wine section

Enter a number: 1

Enter wine name: Verdeca Gravina in Puglia 2018

Wine found!
Title: Verdeca Gravina in Puglia 2018
Type: white
Region: Puglia
Price: 10 €
Taste: crisp
Food: red meat
```

    - Ricerca di un vino nel catalogo tramite il suo ID

```
 1 -- Find by name
 2 -- Find by ID
 3 -- Return to Wine section

Enter a number: 2

Enter wine ID: 11

Wine found!
Title: Tasca d'Almerita 2011 Sallier de la Tour Inzolia
Type: white
Region: Sicily
Price: 13 €
Taste: dry
Food: white meat
```

- **Raccomandazione di un vino**: L'utente definisce le sue preferenze per quanto riguarda le caratteristiche del vino (Tipo, Prezzo, Sapore, Cibo) e il programma raccomanderà il vino corrispondente. C'è inoltre la possibilità di non esprimere una preferenza per una o più caratteristiche del vino, immettendo una semplice "x"

```
 Enter a type [red/white/rose/sparkling] or 'x' for no preference: sparkling
Enter a price or 'x' for no preference: x
Enter a taste [dry/sweet/crisp] or 'x' for no preference: x
Enter a food [red meat/white meat/fish/dessert] or 'x' for no preference: dessert

I found this wine for you!

Title: De Miranda Asti Spumante 2019
Type: sparkling
Region: Piedmont
Price: 18 €
Taste: sweet
Food: dessert
```

- **Aggiunta di un vino**: L'utente può aggiungere un vino non presente nel catalogo, attribuendogli tutte le caratteristiche associate

```
Add your wine in the dataset!

Enter wine's name: Verdeca Gravina in Puglia 2018
Enter wine's type: white
Enter wine's region: Puglia
Enter wine's price: 10
Enter wine's taste: crisp
Enter wine's best food to pair with: red meat

Wine added!
```

5 Considerazioni Finali e Sviluppi Futuri

Il nostro programma pecca sicuramente per quanto riguarda l'automazione nell'aggiornamento della KB, in quanto la conversione dal file .csv non è automatica e la KB è stata riempita manualmente da noi. In futuro si potrebbe implementare un metodo che risolva questo problema, in modo da rendere le basi di conoscenza più ricche di informazioni.

Per quanto riguarda uno sviluppo futuro, il programma potrebbe lavorare su un'unica Knowledge Base, in modo da poter predire la qualità di un vino e aggiungerla come caratteristica da prendere in considerazione per una possibile raccomandazione. Ogni vino, quindi, avrebbe una scheda tecnica per quanto riguarda i composti chimici e una per le caratteristiche fisiche e sensoriali.

Inoltre, mediante l'applicazione di altre misure delle prestazioni e altri algoritmi di machine learning per una migliore comparazione dei risultati, questo studio aiuterà le industrie manifatturiere a prevedere la qualità dei diversi tipi di vino basandosi su determinate caratteristiche e sarà utile anche per loro per produrre un buon prodotto.

Infine, si potrebbe avere una maggiore veridicità dei risultati riorganizzando la KB in modo da distinguere la rappresentazione delle regole in base al tipo di vino. Infatti, i dati dei composti chimici dei vini rossi sono differenti da quelli del vino bianco.