Università degli Studi di Salerno

Dipartimento di Informatica

Sapore DiVino

Progetto realizzato per il corso di *Fondamenti di Intelligenza Artificiale*

Progettista: Fabio Sessa
Matricola: 0512114634

2023/2024

Indice

1 Introduzione

L'universo del vino è un affascinante viaggio sensoriale che coinvolge tutti i nostri sensi, dalla vista al gusto, fino all'olfatto. Gli appassionati di vino apprezzano non solo la bevanda in sé, ma anche la complessità e la raffinatezza che si celano dietro ogni bottiglia. Inoltre, per gli appassionati, risulta essere una sfida sempre più avvincente la ricerca di vini che presentino una ottima qualità e che soddisfino le esigenze. In questo progetto affronteremo quella che è la tematica della qualità dei vini.
Per la realizzazione di questo progetto, presente nella mia repository GitHub, sono presenti tutti i file relativi alla documentazione, presentazione e codice.

1.1 Obiettivi

Durante la realizzazione di questo progetto per il corso Fondamenti di Intelligenza Artificiale, mi sono cimentato in un **problema di classificazione** che riguarda il Machine Learning; riguardo questo campo ho sempre avuto un interesse particolare ma prima di questa esperienza non né ho mai avuto un approccio effettivo.
Le tematiche e gli obiettivi principali sono:

- analisi dettagliata di un dataset di vini rossi;
- l'identificazione di feature associate ai vini;
- l'implementazione di un modello di apprendimento in grado di classificare in modo affidabile la qualità di vini almeno sufficienti e insufficienti.

1.2 Specifica PEAS

- **Performance**: le misure di prestazione adottate per valutare l'operato di un agente. Nel mio caso precision, recall, accuracy e F-score;
- **Environment**: gli elementi che formano l'ambiente. Nel mio caso sarà costituito dall'elenco dei vini rossi;
- **Actuators**: gli attuatori disponibili all'agente per intraprendere le azioni. Nel mio caso sarà l'utilizzo di algoritmi di classificazione;
- **Sensors**: i sensori attraverso i quali l'agente riceve gli input percettivi. Nel mio caso l'agente va ad acquisire i dati utili per classificare la qualità di un vino.

1.3 Caratteristiche dell'ambiente

L'ambiente è:

- **Single-agent**;
- **Parzialmente osservabile**: non si ha accesso a tutte le informazioni complete relative ai vini;
- **Stocastico**: le qualità dei vini possono essere influenzate da fattori difficilmente prevedibili o che presentano una certa variabilità;
- **Sequenziale**: le predizioni passate sulla risposta al trattamento possono fornire un contesto importante per valutare l'efficacia delle qualità successive.
- **Dinamico**: la qualità può variare di vino in vino;
- **Discreto**: le variabili assumono valori in un limitato intervallo; alcune, invece, assumono valori distinti e separati.

1.4 Analisi del problema

L'intero progetto affronta, dunque, quello che un problema di **apprendimento supervisionato** basato sulla costruzione di un modello di Machine Learning. Inoltre, la classificazione che andremo ad utilizzare sarà di tipo **binario**, dato che può assumere valori 0 e 1 (1 se il vino ha una qualità almeno sufficiente, 0 se il vino ha una qualità insufficiente).
Le tecnologie e gli strumenti che ho utilizzato sono i seguenti:

- **Visual Studio Code** come IDE;
- **Python** come linguaggio di programmazione;
- **GitHub** per il versionamento;
- **JupyterNotebook** all'interno dell'IDE;
- **Overleaf** per la scrittura della documentazione;
- **Canva** per la realizzazione della presentazione.

2 Data Understanding

La fase di Data Understanding implica un esame più attento dei dati disponibili per il data mining. Questo passaggio è di fondamentale importanza per evitare problemi imprevisti durante la fase successiva di Data Preparation, che è in genere la parte più lunga di un progetto.
Inoltre questa fase è composta da:

- **Data Collection**;
- **Data Description**;
- **Data Quality**.

2.1 Data Collection

In linea generale, la fase di Data Collection è il processo di raccolta, misurazione e analisi delle informazioni provenienti da innumerevoli fonti diverse. Dopo varie ricerche online del giusto dataset, ho deciso di scegliere un dataset presente su Kaggle, una piattaforma online dedicata alle competizioni di data science, Machine Learning e analisi dei dati. Il riferimento al dataset lo si può trovare al seguente link.

2.2 Data Description

Il dataset scelto presenta un elenco di vini e per gli appassionati, si tratta di vini rossi provenienti dal Portogallo. Possiamo trovare la bellezza di ben 1599 righe e 12 colonne. L'ultima colonna, ovvero *quality*, è quella che utilizzeremo per stabilire se la qualità di un vino è almeno sufficiente (qualità del vino >=6) oppure insufficiente (qualità del vino <6).
Non essendo un esperto ed appassionato di chimica, ho dovuto studiare più nel dettaglio quelle che sono le caratteristiche che formano il dataset. Di seguito vi è una descrizione generale e sintetica delle caratteristiche:

1. **Fixed acidity**: si riferisce alla quantità di acidi presenti nel vino;
2. **Volatile acidity**: si riferisce alla presenza di acidi volatili, ovvero composti chimici acidi che possono evaporare facilmente in forma di gas;
3. **Citric acid**: elemento che può contribuire al profilo di acidità totale del vino;
4. **Residual sugar**: si riferisce agli zuccheri lasciati non fermentati in un vino finito;
5. **Chlorides**: quantità di sale nel vino;
6. **Free sulfur dioxide**: sono i sulfuri disponibili a reagire e quindi presentano proprietà sia germicide che antiossidanti;
7. **Total sulfur dioxide**: rappresenta la somma delle quantità di anidride solforosa libera e combinata presenti nel vino;
8. **Density**: si riferisce alla quantità di materia presente in una determinata quantità di liquido;
9. **pH**: descrive quanto è acido o basico un vino su una scala da 0 (molto acido) a 14 (molto basico);
10. **Sulphates**: si riferiscono all'anidride solforosa (SO2), un composto chimico usato come conservante;
11. **Alcohol**: percentuale di alcol del vino;
12. **Quality**: variabile di output (sulla base dei dati sensoriali, punteggio compreso tra 3 e 8).

Un altro strumento molto utile per capire meglio la distribuzione dei dati è l'uso dei grafici, più in particolare degli istogrammi. Di seguito vi è una rappresentazione grafica riguardante la quantità di alcohol presente nei vini suddivisi per qualità:

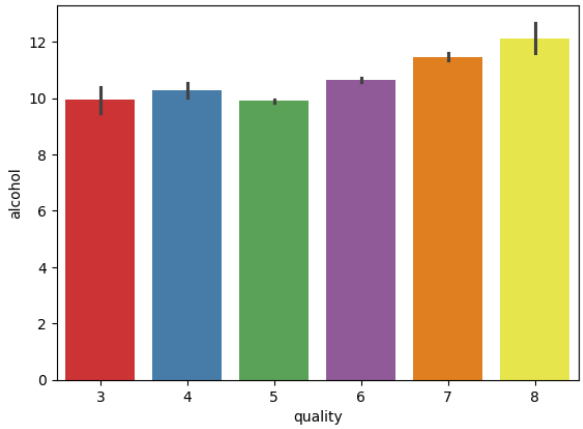

Figura 1: quantità di alcohol nei vini suddivisi per qualità

Come possiamo notare dal grafico, la quantità di alcohol nei vini aumenta con l'aumentare della qualità di quest'ultimi.

Il grafico seguente, invece, rappresenta le frequenze delle qualità:

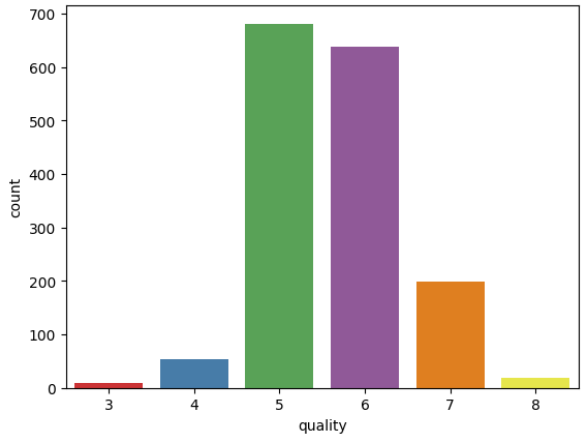

Figura 2: frequenze delle qualità

Salta subito all'occhio che il grafico è meno distribuito rispetto al precedente, ma possiamo trarre alcune conclusioni, come ad esempio la moda della qualità che è rappresentata dal valore 5; inoltre la quantità di vini con qualità ottima/eccellente è molto scarsa.

2.3 Data Quality

In questa fase vado ad analizzare la qualità dei miei dati. L'obiettivo primario, ricordiamo, è la corretta classificazione della qualità di un vino e classificarla come almeno sufficiente (assegnando il valore **1**) o come insufficiente (assegnando il valore **0**). Non mi resta che suddividere la qualità in queste due categorie e con l'aiuto del grafico possiamo studiare al meglio la situazione:

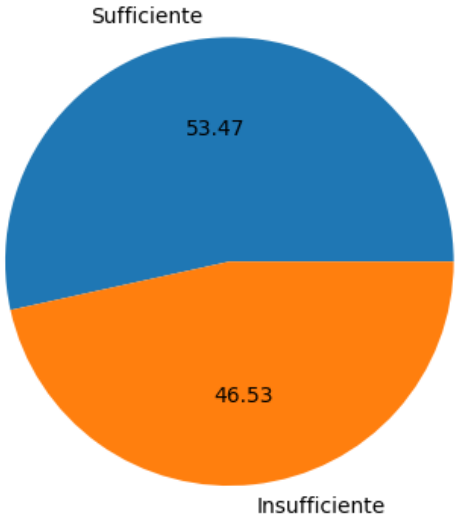

Figura 3: percentuali di frequenza delle classi

Dal grafico a torta possiamo notare che i vini con una qualità almeno sufficiente sono molti di più rispetto ai vini con una qualità insufficiente. Inoltre, dobbiamo gestire anche il problema legato allo sbilanciamento, ma questa tematica sarà affrontata più avanti nella fase di Data Balancing.

3 Data Preparation

La fase di Data Preparation ha lo scopo di eseguire una serie di attività volte a raccogliere, pulire, trasformare e organizzare i dati in modo che siano pronti per l'analisi o per l'addestramento di modelli di Machine Learning. Tale fase è suddivisa in diverse sotto-fasi, eseguite in modo cronologico:

- **Data Cleaning**;
- **Feature Scaling**;
- **Feature Selection**;
- **Data Balancing**.

3.1 Data Cleaning

La fase di Data Cleaning è progettata per identificare e correggere gli errori, le inconsistenze e le irregolarità nei dati raccolti. Alcuni esempi possono essere la presenza di dati mancanti in alcune righe, la presenza di intere colonne con valore nullo oppure la presenza di duplicati all'interno del dataset. Nel mio caso, non avendo valori nulli, ho dovuto affrontare solo il problema dei duplicati (ben 240) che non ho esitato a rimuovere, dato che potrebbero creare problemi nella classificazione binaria.

3.2 Feature Scaling

La fase di Feature Scaling consiste nel trasformare le variabili del dataset in modo che siano tutte su una scala comune. Il motivo principale dietro tale fase è che molte tecniche di Machine Learning possono essere influenzate negativamente da differenze nelle scale delle variabili.
Le due tecniche che abbiamo studiato sono: min-max normalization e z-score normalization; tra le due ho optato per la tecnica del **min-max normalization**, dove l'obiettivo di questa tecnica è trasformare le variabili in modo che abbiano valori compresi tra un certo intervallo, di solito tra 0 e 1.
Prima di proseguire con la normalizzazione, mi sono adoperato ad effettuare la divisione del dataset e affrontare il problema del *data leakage* (fuga di dati). La divisione del dataset è avvenuta in due gruppi: il 70% del è dedicato all'addestramento (**training set**) e il 30% è dedicato al testing (**test set**). Dopo aver normalizzato posso passare alla fase successiva, ossia la Feature Selection.

3.3 Feature Selection

Tale fase è un processo che coinvolge la scelta delle variabili (o caratteristiche) più rilevanti da utilizzare per costruire un modello predittivo o analizzare i dati. In generale, le *feature* si riferiscono alle variabili o attributi che vengono utilizzati per descrivere gli esempi nei dati.
Per ottenere una panoramica generale delle correlazioni tra le varie caratteristiche, mi sono aiutato con quella che viene chiamata **matrice di correlazione**: una matrice quadrata in cui ogni cella rappresenta la correlazione tra due variabili. Di seguito una guida per leggere al meglio una matrice di correlaizione:

- la diagonale principale della matrice contiene sempre valori di correlazione 1, poiché una variabile è sempre perfettamente correlata con se stessa;
- le celle più scure indicano correlazioni più forti;
- le variabili sono fortemente correlate tra loro se hanno valori vicini, in genere, a 1 o -1, mentre hanno una bassa correlazione se hanno valori vicini a 0.

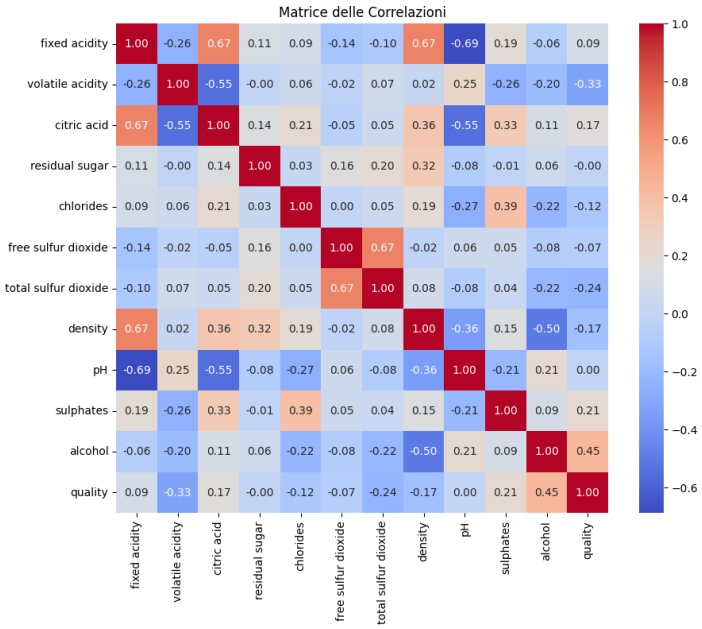

Figura 4: matrice delle correlazioni

La matrice di correlazione appena calcolata presenta il problema di **multicollinearità**, ovvero in cui due o più variabili indipendenti sono fortemente correlate tra loro; infatti:

- **free-sulfur-dioxide** e **total-sulfur-dioxide** sono ben correlate positivamente (0.67);
- **fixed-acidity** è ben correlata positivamente (0.67) con **density** e **citric-acid**;
- **pH** e **fixed-acidity** sono altamente correlate negativamente (-0.69).

Per affrontare il problema di multicollinearità, decido di scartare le feature **fixed-acidity**, **citric-acid** e **total-sulfur-dioxide**.

3.4 Data Balancing

L'obiettivo del Data Balancing è affrontare situazioni in cui le classi di un insieme di dati non sono rappresentate in modo uniforme, cioè alcune classi possono essere sovrarappresentate rispetto ad altre. Come enunciato in precedenza, il mio dataset presenta una classe maggioritaria di vini con qualità almeno sufficiente rispetto a vini con qualità insufficiente. Le tecniche che ho studiato durante il corso sono:

- la tecnica dell'**under-sampling**: eliminare casualmente le istanze della classe di maggioranza;
- la tecnica dell'**over-sampling**: aggiungere casualmente le istanze della classe di minoranza.

Entrambe le tecniche vanno usate nel giusto modo e trattate con particolare attenzione.
Per bilanciare il dataset e per evitare di andare ad effettuare duplicazioni che potrebbero causare overfitting (quando un modello di Machine Learning si adatta troppo bene ai dati di addestramento) con l'utilizzo dell'over-sampling, ho preferito utilizzare la tecnica dell'**under-sampling**.
La rimozione delle istanze della classe maggioritaria è stata effettuata tramite *RandomUndersampler*, uno strumento contenuto nella libreria sklearn.

4 Data Modeling

Dopo aver effettuato le varie operazioni riguardanti la Data Preparation, possiamo affrontare quella che è la fase del Data Modeling. Tale fase, infatti, comprende diverse attività cruciali che si basano sulla rappresentazione dei dati in modo che possano essere utilizzati efficacemente dal modello di Machine Learning.

4.1 Scelta dei classificatori

I classificatori sono algoritmi o modelli che vengono addestrati per assegnare etichette ai dati, sulla base delle informazioni apprese da dati precedentemente etichettati durante la fase di addestramento. Nel mio caso, l'intero progetto è strutturato su un problema di classificazione binaria, dove la variabile target può assumere può assumere il valore 0 se la qualità è insufficiente, oppure il valore 1 se la qualità è almeno sufficiente. Tra i classificatori studiati durante il corso, possiamo trovare l'algoritmo di **Naive Bayes** (e relative varianti) e i **decision tree** (alberi decisionali). Navigando online se né possono trovare a migliaia, tra cui il Random Forest, K-Nearest Neighbors etc.
La mia scelta di implementazione dei classificatori è ricaduta su:

- Naive Bayes (con relative varianti);
- Decision Tree;
- Random Forest.

4.1.1 Naive Bayes (definizione)

L'algoritmo di Naive Bayes è un tipo di algoritmo di apprendimento automatico supervisionato basato sul teorema di Bayes, che sfrutta la probabilità condizionale. L'approccio "naive" (ingenuo) deriva dal fatto che l'algoritmo assume l'indipendenza condizionale tra ogni coppia di caratteristiche, anche se questo potrebbe non essere sempre realistico nella pratica. Inoltre all'algoritmo di Naive Bayes sono associate diverse varianti, che però daremo la definizione del loro comportamento nella fase di addestramento; per adesso mi limito solo ad elencarle:

- Gaussian Naive Bayes;
- Multinomial Naive Bayes;
- Bernoulli Naive Bayes.

4.1.2 Decision Tree (definizione)

Gli alberi decisionali sono modelli utilizzati per compiti di classificazione (anche di regressione). Sono costruiti a partire da un set di dati di addestramento, e il loro obiettivo è quello di apprendere regole decisionali basate sugli attributi dei dati. Un albero decisionale è costituito da **nodi** e **archi**. Ogni nodo rappresenta una decisione basata su un attributo specifico, mentre gli archi collegano i nodi e rappresentano le possibili scelte o risultati delle decisioni. Il nodo iniziale, chiamato nodo radice, rappresenta la decisione iniziale basata sull'attributo più significativo, attraverso misure come l'entropia o l'information-gain. Gli altri nodi, noti come nodi interni, rappresentano decisioni intermedie, mentre i nodi foglia rappresentano le etichette di classificazione.

4.1.3 Random Forest (definizione)

Il Random Forest è un algoritmo di Machine Learning che rientra nella categoria degli algoritmi di *ensemble learning*. Gli ensemble learning combinano diversi modelli per migliorare le prestazioni complessive del sistema. Il Random Forest è particolarmente efficace per problemi di classificazione come in questo caso.

4.2 Lettura di una matrice di confusione

Successivamente ci sarà utile la giusta lettura di un matrice di confusione: essa è una matrice che riassume le istanze di test in base ai valori predetti e ai valori veri; essi sono presentati come tabella di contingenza:

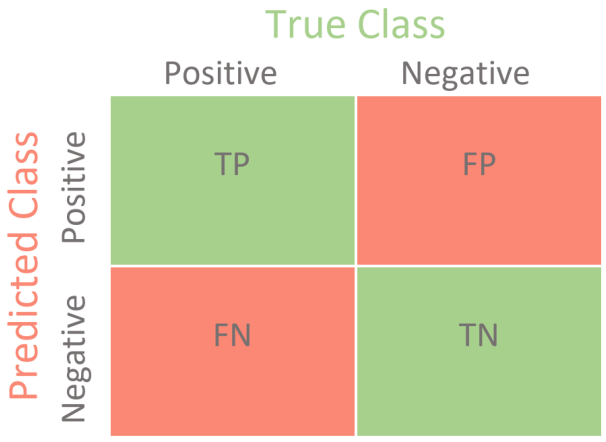

Figura 5: matrice di confusione

Rispettivamente abbiamo:

- **True Positive** (TP, in alto a sinistra): il modello ha risposto correttamente *almeno sufficiente*;
- **True Nevative** (TN, in basso a destra): il modello ha risposto correttamente *insufficiente*;
- **False Positive** (FP, in alto a destra): il modello ha risposto in modo errato *almeno insufficiente* invece di *insufficiente*;
- **False Negative** (FN, in basso a sinistra): il modello ha risposto in modo errato *insufficiente* invece di *almeno sufficiente*.

4.3 Addestramento

In questa sezione procedo ad addestrare il modello in base agli algoritmi scelti in precedenza. Nel mio caso, come già detto, procedo prima all'addestramento usando Naive Bayes, per poi fare lo stesso con Decision Tree e infine Random Forest. Dopodiché valuterò le prestazioni ottenute e le metterò a confronto. Tutto ciò mi consentirà di capire quale algoritmo si adatta meglio al problema, e va di conseguenza a fare delle predizioni migliori.

4.3.1 Gaussian Naive Bayes

L'algoritmo Gaussian Naive Bayes è una delle varianti dell'algoritmo di Naive Bayes ed è particolarmente utile quando si lavora con dati che possono essere descritti utilizzando distribuzioni gaussiane (normali). Andiamo a studiare più nel dettaglio il comportamento di questo algoritmo:

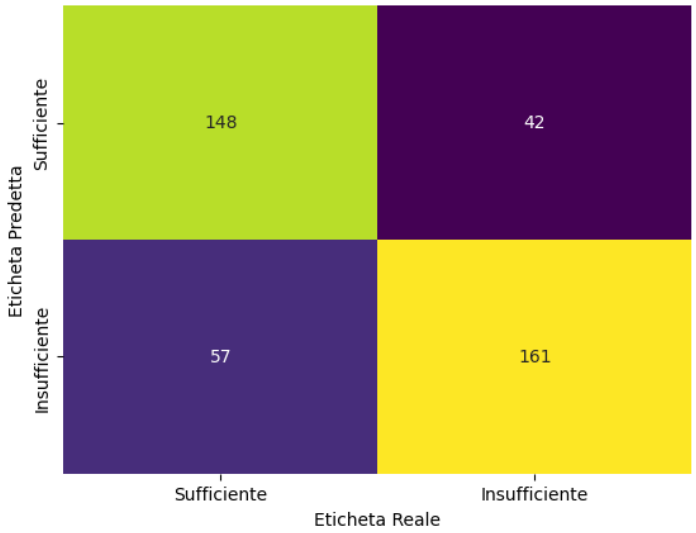

Figura 6: matrice di confusione Gaussian Naive Bayes

4.3.2 Multinomial Naive Bayes

L'algoritmo Multinomial Naive Bayes è la seconda variante che andremo a studiare dell'algoritmo di Naive Bayes ed è particolarmente utile per problemi di classificazione di testi.
Notiamo la matrice di confusione disegnata:

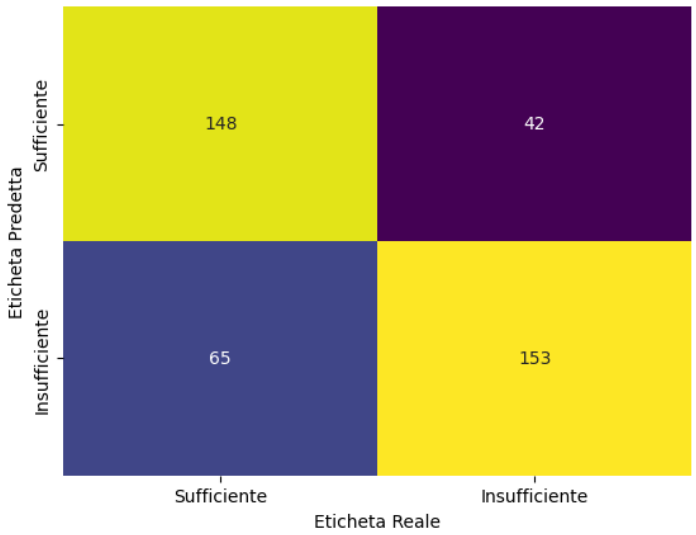

Figura 7: matrice di confusione Multinomial Naive Bayes

4.3.3 Bernoulli Naive Bayes

Infine, l'ultima variante che andremo a studiare è quella relativa all'algoritmo Bernoulli Naive Bayes. Tale algoritmo è particolarmente utile per dati binari. Vediamo come si comporta:

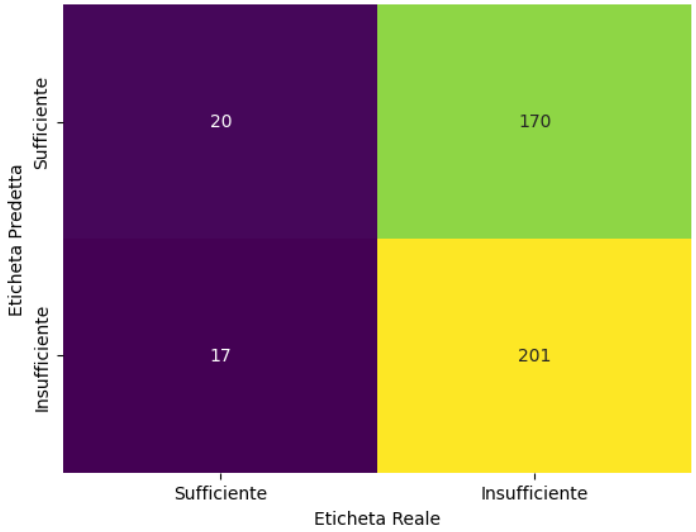

Figura 8: matrice di confusione Bernoulli Naive Bayes

4.3.4 Confronto tra GNB - MNB - BNB

Dopo aver studiato il comportamento di ogni singola variante dell'algoritmo di Naive Bayes, passiamo a confrontare i risultati ottenuti e fare le prime deduzioni. Per fare ciò ho bisogno di metriche specifiche, chiamate **metriche di valutazione**. Le principaili e quelle più famose sono **precision**, **recall** e **accuracy**, ma ho voluto studiare il comportamento anche della metrica **F-score**:

- **Precision**: misura la proporzione di casi positivi effettivamente positivi tra tutte le predizioni positive fatte dal modello;
- **Recall**: misura la capacità del modello di identificare correttamente i casi positivi rispetto al totale dei casi positivi reali;
- **Accuracy**: misura la percentuale di tutte le predizioni corrette rispetto al numero totale di predizioni;
- **F-score**: misura che viene calcolata tramite la media armonica di precision e recall.

Classificatore	Precision	Recall	Accuracy	F-score
Gaussian NB	0.793103	0.738532	0.757353	0.764846
Multinomial NB	0.784615	0.701835	0.737745	0.740920
Bernoulli NB	0.541779	0.922018	0.541667	0.682513

Come possiamo notare dalla tabella, i valori delle metriche non sono altissimi: in particolare notiamo che l'algoritmo Bernoulli Naive Bayes pecca sulla precision e sull'accuracy ma compensa con un valore di recall abbastanza alto; per quanto riguarda gli altri due algoritmi hanno risultati simili. Per scelta implementativa, l'obiettivo è quello di **massimizzare la metrica F-score** e l'algoritmo massimizza questa metrica è l'algoritmo di **Gaussian Naive Bayes** (0.764846).

4.3.5 Decision Tree

Dopo aver dato la definizione in precedenza, cerchiamo di implementare l'algoritmo di Decision Tree, studiamo come si adatta ai dati e vediamo se otteniamo miglioramenti rispetto agli algoritmi di Naive Bayes:

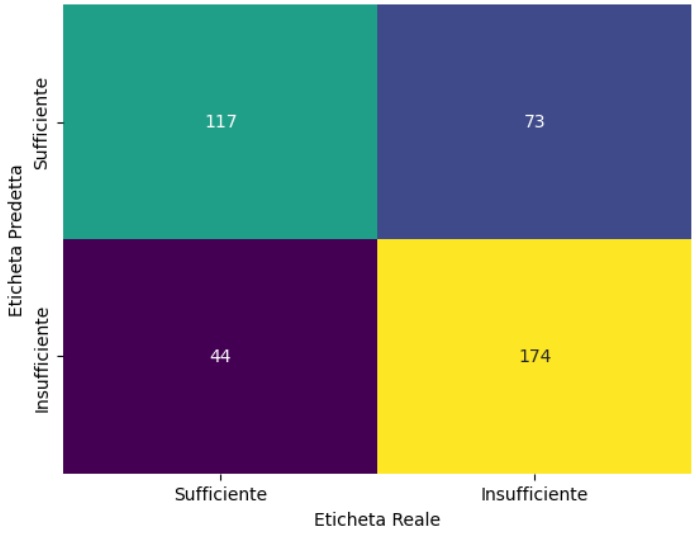

Figura 9: matrice di confusione Decision Tree

Classificatore	Precision	Recall	Accuracy	F-score
Decision Tree	0.704453	0.798165	0.713235	0.748387

Rispetto all'algoritmo Gaussian Naive Bayes (0.764846) abbiamo avuto un **peggioramento dell'F-score**.

4.3.6 Random Forest

L'ultimo algoritmo che sono andato ad implementare è Random Forest. Vediamo come si comporta:

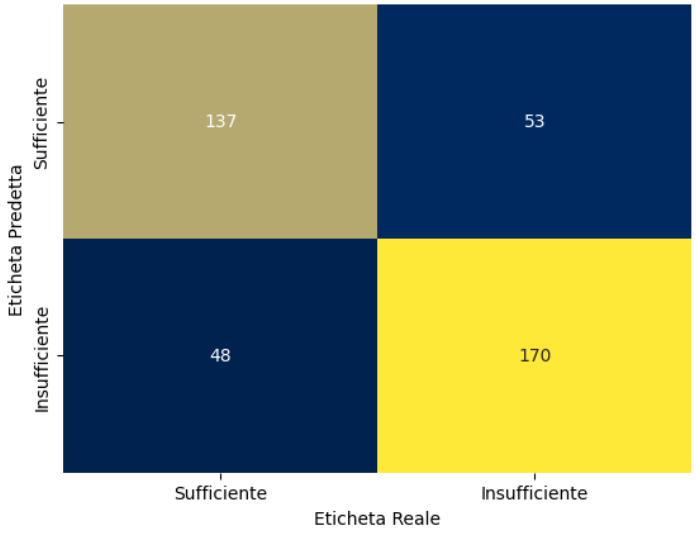

Figura 10: matrice di confusione Random Forest

Classificatore	Precision	Recall	Accuracy	F-score
Random Forest	0.762332	0.779817	0.752451	0.770975

In questo caso, abbiamo avuto un miglioramento della metrica F-score rispetto alla precedente del Gaussian Naive Bayes (0.764846).

4.4 Risultati

Implementati tutti quelli che sono gli algoritmi scelti per la classificazione, facciamo una panoramica e mettiamoli tutti a confronto tramite l'uso di un istogramma:

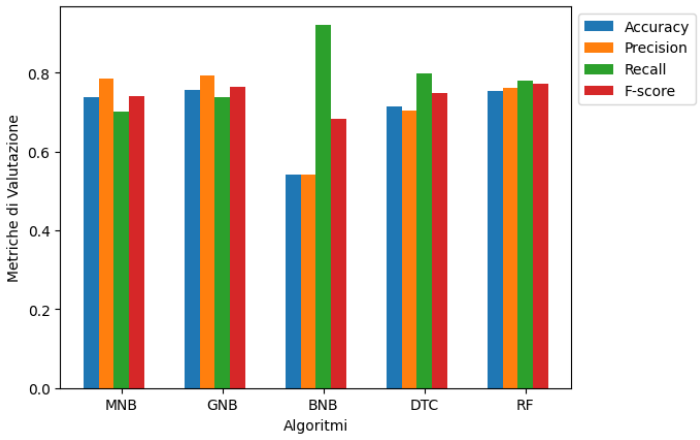

Figura 11: confronto delle metriche di ogni algoritmo implementato

Possiamo notare banalmente che alcuni algoritmi (come MNB, GNB) operano davvero in modo molto simile oppure che la metrica di recall è sfruttata al meglio con l'algoritmo Bernoulli Naive Bayes. Vediamo in modo specifico il confronto tramite l'uso di una tabella

Classificatore	**Precision**	**Recall**	**Accuracy**	**F-score**
Gaussian NB	0.793103	0.738532	0.757353	0.764846
Multinomial NB	0.784615	0.701835	0.737745	0.740920
Bernoulli NB	0.541779	0.922018	0.541667	0.682513
Decision Tree	0.704453	0.798165	0.713235	0.748387
Random Forest	0.762332	0.779817	0.752451	**0.770975**

Come già detto in precedenza, l'obiettivo è quello di **massimizzare il valore della metrica F-score** (quindi avere il miglior apporto precision-recall) e l'algoritmo che si adatta meglio al valore è quello del **Random Forest**.

5 Evaluation

Per avere più chiara la bontà del modello ho effettuato un'ulteriore valutazione attraverso la **ROC curve** (legata al Random Forest), che consente di visualizzare il trade-off tra recall e specifity. In poche parole misura la capacità di identificare correttamente i casi negativi, ovvero quanto bene il modello riesce a distinguere tra vini di con qualità insufficiente o almeno sufficiente:

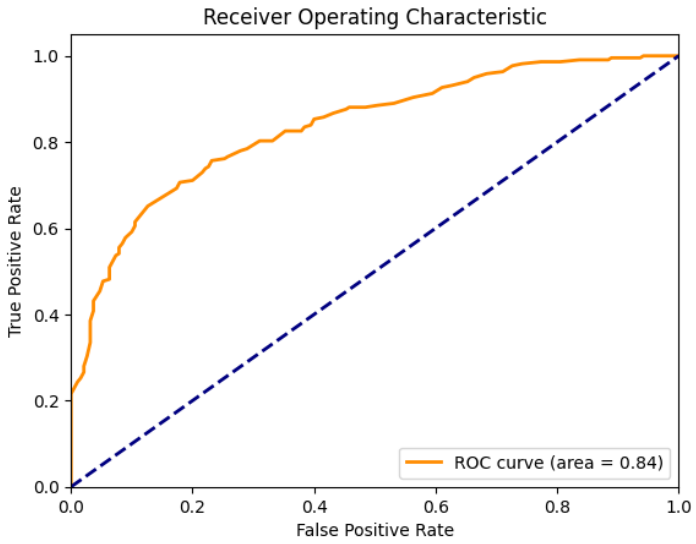

Figura 12: curva ROC - AUC

La linea diagonale rappresenta il risultato di una scelta casuale, quindi un qualunque modello utile deve essere al suo di sopra. Inoltre un buon modello deve avere sensibilità elevata e basso tasso di falsi positivi, cioè il modello deve riuscire correttamente a predirre che un vino abbia una qualità almeno sufficiente senza generare falsi positivi, ovvero predirre che un vino abbia una qualità almeno sufficiente, ma in realtà non lo ha.

Infine, la **AUC** (Area sotto la curva) che fornisce una misura generale delle prestazioni del modello, ad esempio un'area di 1.0 indica un modello perfetto, mentre un'area di 0.5 indica una scelta casuale. Il mio modello ha ottenuto un valore pari a 0.84 che è non è eccellente, ma un ottimo risultato. Quindi posso considerare la costruzione del modello e in generale dell'approccio completa!

6 Conclusioni

I risultati ottenuti sono ottimi ma non perfetti: ciò potrebbe essere legato al fatto che il dataset preso in considerazione presenti la sola descrizione degli elementi chimici di cui è composto un vino, ma per determinare la qualità di un vino come ottima avremmo bisogno anche di informazioni come la varietà d'uva, la zona di provenienza, il grado di maturazione e così via.
Tutto sommato, la realizzazione di questo progetto mi ha portato ad avere un primo approccio su un problema di Machine Learning. Posso dire che è stata un'esperienza che mi ha divertito e soprattutto appassionato nella ricerca di effettuare miglioramenti al modello. Che questo possa essere solo l'inizio di una strada interessante come quella dell'Intelligenza Artificiale!