# Red wine quality classifier

Sommario



## 1. Introduzione

Il supervised learning è il problema più studiato nel machine learning. Esso si pone l'obiettivo di prevedere, dato un elemento di cui si conoscono un insieme di parametri (features), il valore di un diverso parametro di output relativo all'elemento stesso. Per far ciò, nel supervised learning viene definito (mediante apprendimento da insiemi di esempi) un modello.

Il seguente caso di studio consiste nel confronto e nella valutazione di cinque modelli di classificazione basati su apprendimento supervisionato, in particolare quello che si vuole classificare è la qualità del vino rosso, dati dei parametri chimico-fisici.

I modelli di classificazione confrontati sono stati scelti fra i principali delle seguenti categorie

di apprendimento:

- Case Based Reasoning: K-Nearest Neighbors Classifier
- Probabilistic Classifiers: Naïve Bayes Classifier
- Ensemble Learning Model: Random Forest Classifier
- Ensemble Learning Model with boosting: AdaBoost Classifier
- Linear model: Support Vector Machine Classifier

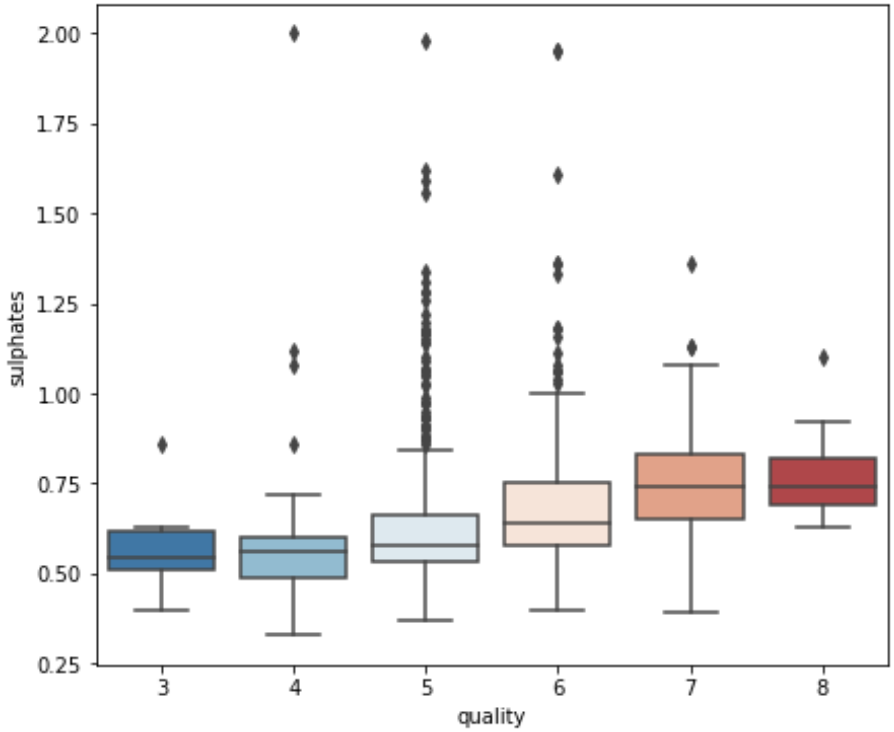
## 2. Dataset

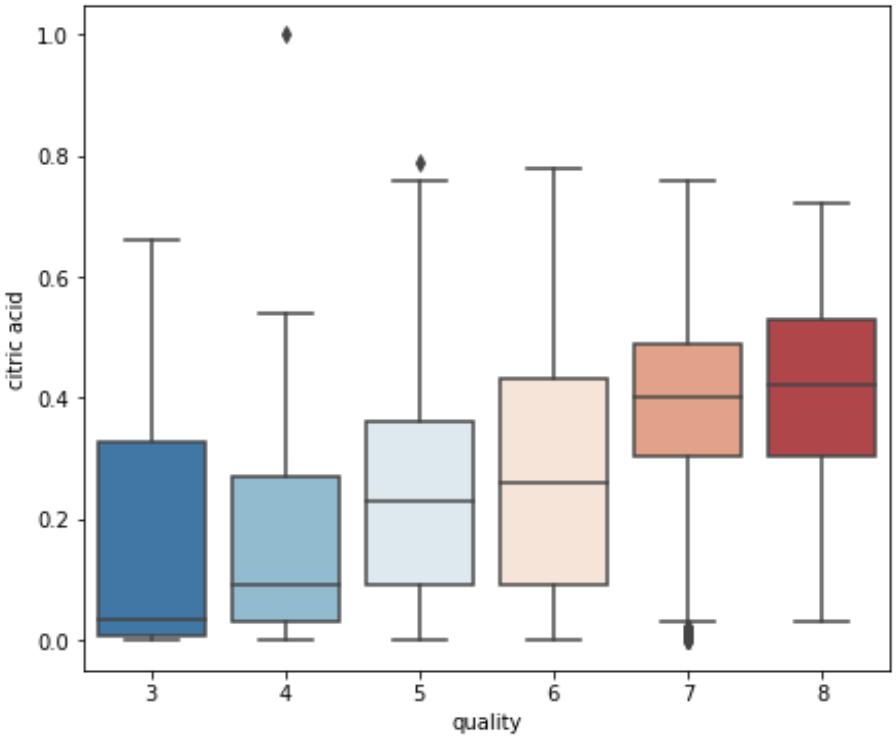
Il dataset utilizzato appartiene alla repository di UCI. Nella repository vi sono due set di dati relativi alle varianti rossa e bianca del vino portoghese "Vinho Verde", in questo progetto ci si è interessati unicamente al a quello di vino rosso. Il dataset presenta 1599 esempi di vini diversi di cui sono omessi i nomi per ragioni di privacy. Le feature di input sono le seguenti:

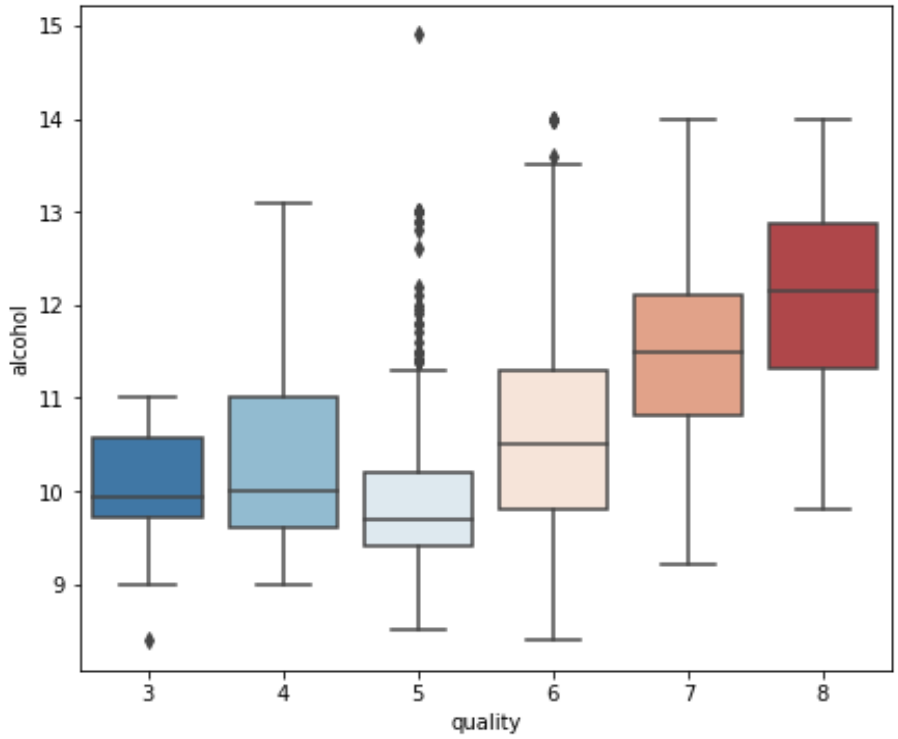
- **Fixed acidity** (**acidità fissa**): acidità dovuta a tutti gli altri acidi (non quella volatile) che non si disperdono durante la vita di un vino.
- **Volatile acidity** (**acidità volatile):** la quantità di acido acetico nel vino, che ha la possibilità di liberarsi volatilizzandosi. A livelli troppo alti può portare a un sapore sgradevole.
- **Citric acid** (**acido citrico**) in piccolo quantità rende il vino più “fresco”
- **Residual sugar** (**zucchero residuo**): la quantità di zucchero rimanente dopo la fine della fermentazione, è raro trovare vini con meno di 1 grammo / litro e vini con più di 45 grammi / litro questi sono considerati dolci;
- **Chlorides** (**cloruri**): la quantità di sale nel vino;
- **Free sulfur dioxide** (**Anidride solforosa libera**): previene la crescita microbica e l'ossidazione del vino;
- **Total sulfur dioxide** (**anidride solforosa totale**): quantità di forme libere e legate di S02; a basse concentrazioni SO2 è per lo più non rilevabile nel vino, ma a concentrazioni di SO2 libera superiori a 50 ppm, SO2 diventa evidente e nel gusto e nell’odore del vino;
- **Density** (**densità**): è vicina a quella dell'acqua a seconda della percentuale di alcol e del contenuto di zucchero;
- **pH:** descrive quanto è acido o basico un vino su una scala da 0 (molto acido) a 14 (molto basico); la maggior parte dei vini è compresa tra 3-4 sulla scala del pH;
- **Sulphates** (**Solfati**) un additivo per vino che può contribuire ai livelli di anidride solforosa (S02), che agisce come un antimicrobico e antiossidante;
- **Alcohol:** la percentuale di contenuto alcolico del vino;;
- La feature di output è:
- **Quality**: la qualità su una scala da 1 a 10

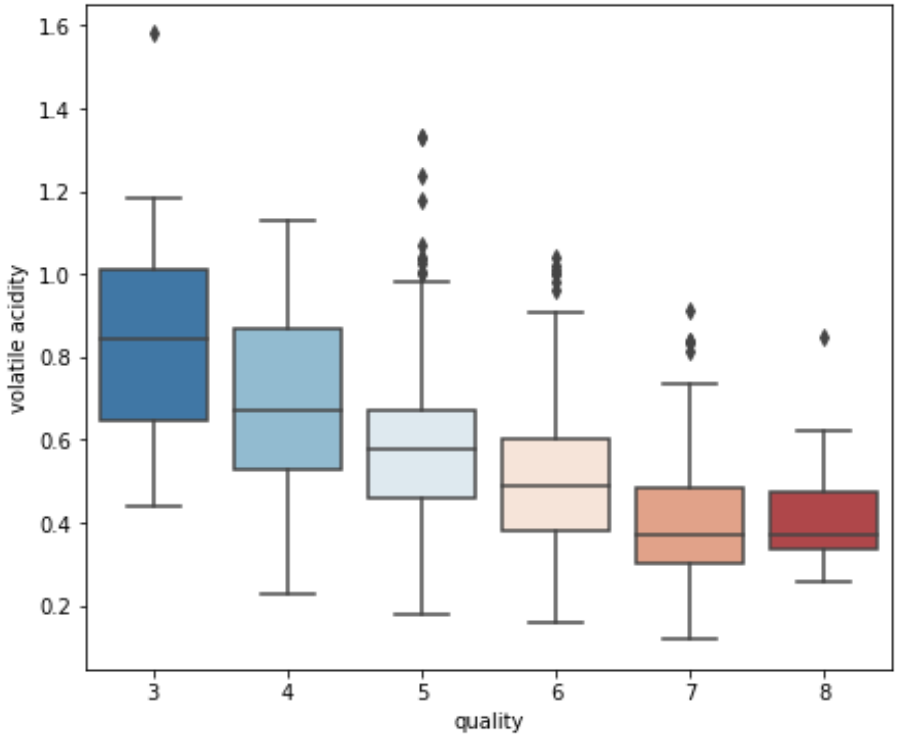
2.1.Correlazione tra feature di output e feature di input

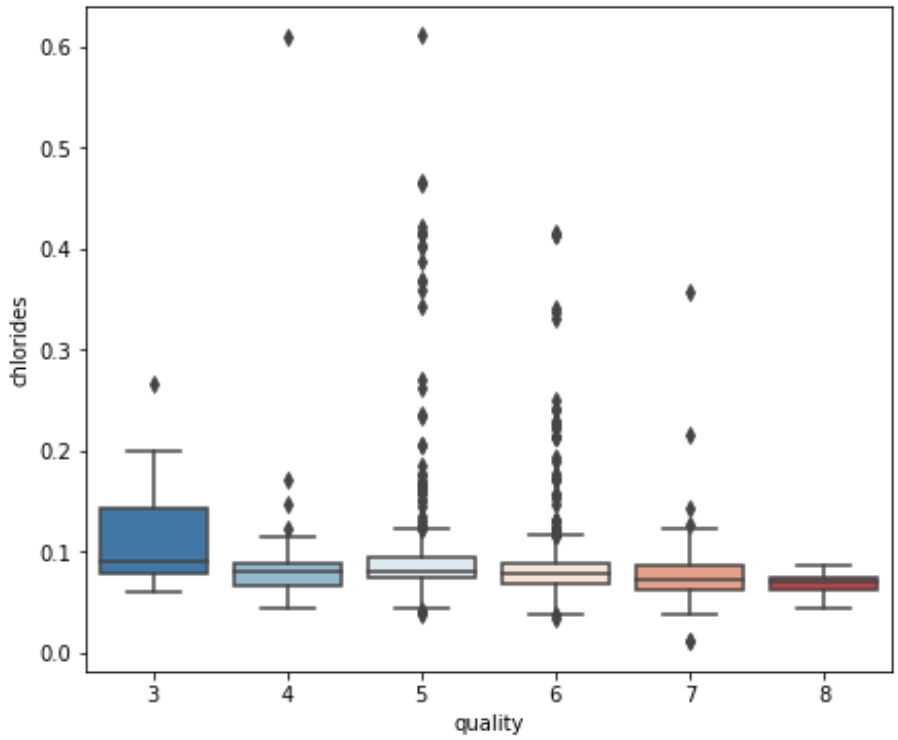
Prima di procedere con il task di classificazione si è analizzato la correlazione tra le feature di input e la feature obiettivo, di modo che fosse possibile individuare quale tra le componenti influenzano maggiormente la qualità del vino. Mediante l'utilizzo delle librerie seaborn e matplotlib sono stati creati boxplot per visualizzare la correlazione tra la quality e le altre feature.




alcohol
15
14
13
12
11
10
9
3
4
5
6
7
8
quality

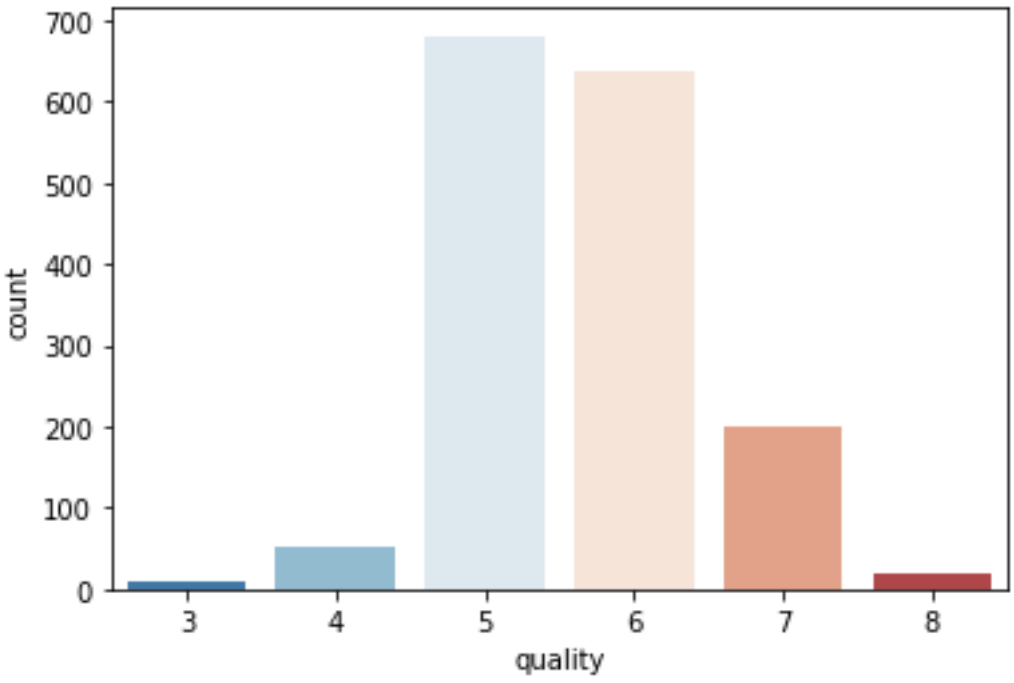

volatile acidity
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
3
4
5
6
7
8
quality



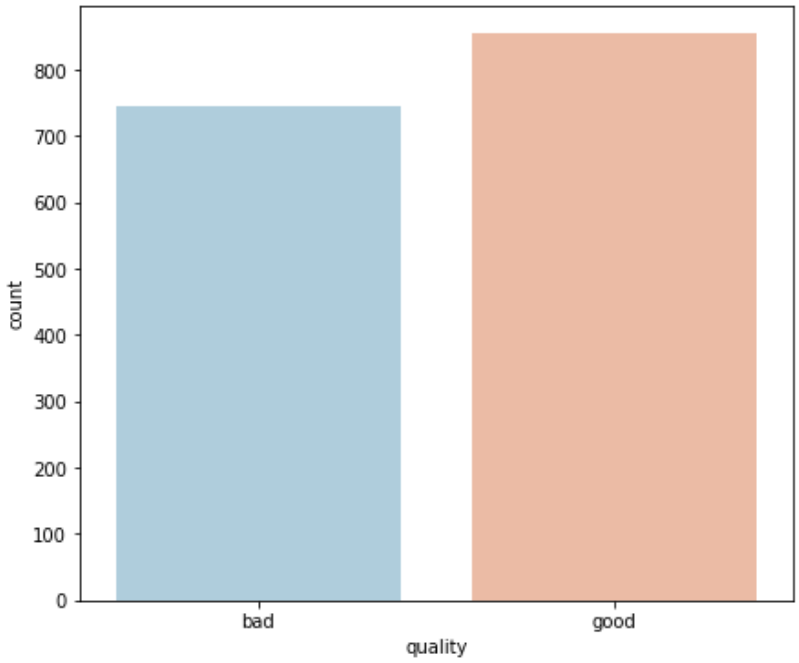

I boxplot riportati sono quelli più significativi, si possono notare i seguenti effetti delle varie feature di input sulla qualità: all'aumento dell' acido citrico corrisponde un aumento della qualità, lo stesso vale per i solfati e l'alcohol, mentre con l'aumento dei cloridi e gli acidi volatili la qualità diminuisce.

## 2.2.Visualizzazione del dataset e preprocessing

Il dataset non presenta valori Nan e sono tutti valori numerici, in compenso risulta molto sbilanciato.



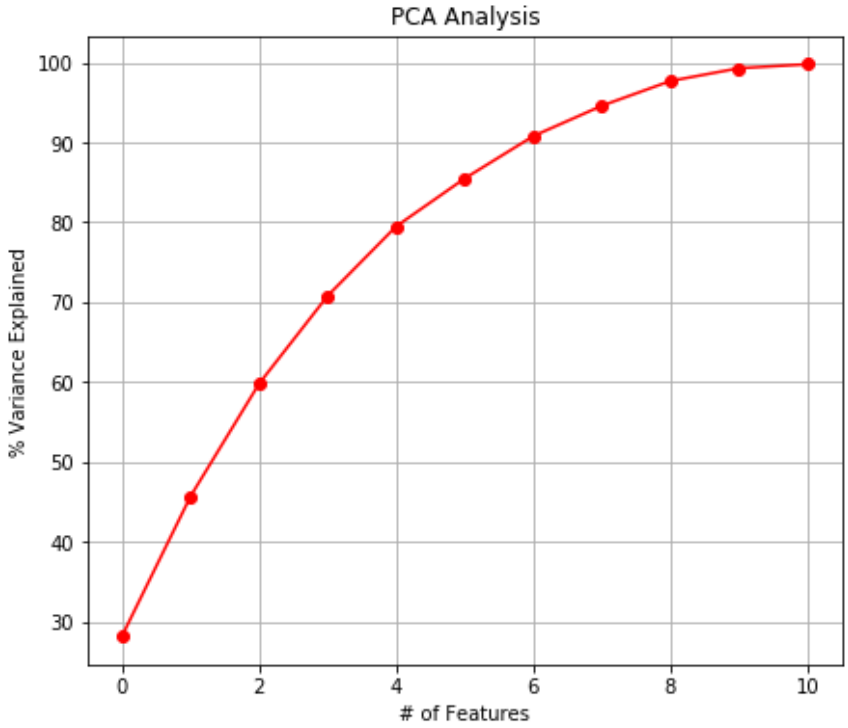
Come si evince da grafico vi sono la maggior parte dei valori per la feature target sono uguali a 5 e 6. Per questo si sono raggruppati le votazioni di qualità in due sottoinsiemi: i vini classificati con valori minori di 5.5 sono stati etichettati con “bad” i restanti con “good”.




2.3. PCA

L'analisi delle componenti principali è una tecnica utilizzata per ridurre la dimensionalità di un set di dati. La PCA è stata impiegata perché riduce al minimo il numero di variabili utilizzate. Questa utilizza la "trasformazione lineare ortogonale" per proiettare le feature su un nuovo sistema di coordinate in cui l'elemento che ha la maggior varianza diviene la prima coordinata (diventando così il primo componente principale).

Per eseguire la PCA sono stati standardizzati i dati, utilizzati per creare la matrice delle covarianze, la quale, a sua volta, è stata utilizzata per calcolare gli autovalori (le componenti principali) e i rispettivi autovettori. Si sono ordinate le componente e scelte quelle con più varianza (dato che quelle con più varianza descrivono meglio i dati). Infine si è creato una nuova matrice con le nuove componenti.




Nel grafico precedente è visibile come le prime otto componenti sono quelle che esprimono maggior varianza, mentre dall'ottava in poi c'è un guadagno di informazione poco significativo. Sono state selezionate le prime otto componenti.

## 3. Classificazione

Utilizzando la libreria sklearn sono stati costruiti diversi modelli di classificazione.

**Support Vector Machine:**

il Support Vector Machine ha l'obiettivo di identificare l'iperpiano che meglio divide i vettori di supporto in classi. Per farlo esegue i seguenti step:

Cerca un iperpiano linearmente separabile o un limite di decisione che separa i valori di una classe dall'altro. Se ne esiste più di uno, cerca quello che ha margine più alto con i vettori di supporto, per migliorare l'accuratezza del modello.

Se tale iperpiano non esiste, SVM utilizza una mappatura non lineare per trasformare i dati di allenamento in una dimensione superiore (se siamo a due dimensioni, valuterà i dati in 3 dimensioni). In questo modo, i dati di due classi possono sempre essere separati da un iperpiano, che sarà scelto per la suddivisione dei dati.

I nuovi esempi sono mappati nell'iperpiano e la predizione della categoria alla quale appartengono viene fatta individuando il lato dell'iperpiano nel quale ricade.

L'algoritmo SVM ottiene la massima efficacia nei problemi di classificazione binari.

**Random Forest:**

È un modello ottenuto dall'aggregazione tramite bagging di alberi di decisione. Esso è un meta-stimatore che si adatta ad una serie di alberi decisionali addestrati su vari sotto-campioni del dataset e utilizza la media di ogni singolo output di ogni albero per migliorare l'accuratezza predittiva e il

controllo del sovradattamento. Il Random Forest deve essere dotato di due matrici: una matrice X sparsa che contiene i campioni di addestramento e una matrice Y di dimensioni che contiene i valori target.

**GaussianNB:**

l'algoritmo supporta dati numerici continui e presuppone che i valori di ciascuna caratteristica siano normalmente distribuiti (ossia ricadono da qualche parte su una curva a campana). In altre parole, Naive Bayes può essere esteso ad attributi a valori reali, più comunemente assumendo una distribuzione gaussiana o normale. Secondo questa assunzione è sufficiente trovare la media e la deviazione standard di ciascuna probabilità per ogni attributo e per ogni singola classe. Sostituendo tali valori nella funzione di densità di probabilità gaussiana (detta anche Gaussian Probability Density Function) si ricava una probabilità che permette di ricavare le varie probabilità di classe. Il valore di probabilità di classe più alto così ottenuto rappresenta la classe da associare alla nuova istanza che si vuole categorizzare.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

**K-Nearest Neighbors:**

Viene chiamato algoritmo lazy learner perché non apprende immediatamente dal set di addestramento, ma memorizza il set di dati e al momento della classificazione esegue un'azione sul set di dati. Infatti, calcola la somiglianza tra un nuovo esempio e gli esempi disponibili nel dataset assegnandone l'etichetta più simile alle categorie disponibili. In altre parole, memorizza tutti i dati disponibili e classifica un nuovo esempio in base alla somiglianza.

**AdaBoost:**

AdaBoost è un modello di ensemble boosting che utilizza alberi decisionale. L'output del meta-classificatore (alberi decisionali) è dato dalla somma pesata delle predizioni dei singoli modelli. Ogni qual volta un modello viene addestrato, ci sarà una fase di ripesaggio delle istanze. L'algoritmo di boosting tenderà a dare un peso maggiore alle istanze misclassificate, nella speranza che il successivo modello sia più esperto su quest'ultime. Sostanzialmente, ad ogni iterata calcola il tasso di errore ponderato dell'albero decisionale, ovvero il numero di predizioni sbagliate sul totale delle predizioni, che dipende dai pesi associati agli esempi nel dataset; successivamente in base all'errore calcola il learning rate dell'albero decisionale.

maggiore è il tasso di errore di un albero, minore sarà il potere decisionale che l'albero avrà durante la predizione successiva

minore è il tasso di errore di un albero, maggiore sarà il potere decisionale assegnato all'albero durante la predizione successiva.

## 3.1. Iperparametri

Mediante *GridSearchCV,* si sono generati in maniera esaustiva i possibili candidati (iperparametri) attraverso una griglia di valori specificata opportunamente dal parametro *"param_grid",* caratterizzato da un range di valori per ogni singolo parametro specificato dall'utente. In maniera del tutto automatica, vengono valutate tutte le possibili combinazioni di assegnazioni degli iperparametri e viene mantenuta la combinazione migliore. Al termine di tale processo, verranno mostrati quelli che sono gli iperparametri migliori per un determinato modello di classificazione.

Gli iperparametri utilizzati sono:

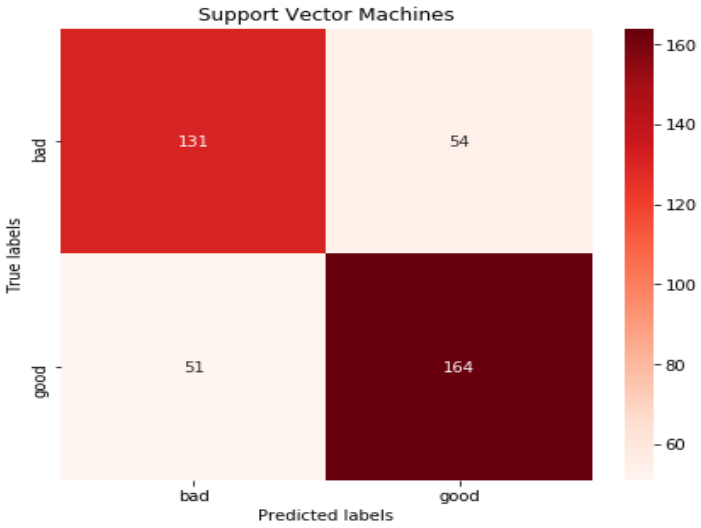
- C: 0.1, 1, 10, 100, 1000; gamma: 1, 0.1, 0.01, 0.001, 0.0001; kernel: rbf, sigmoid; per SVM
- n_estimators: 100, 250, 500; max_features: auto, log2; criterion :gini, entropy per RF
- n_neighbors:1,4,5,6,7,8; leaf_size:1,3,5,10; per KNN

I migliori sono stati

- C: 10, gamma: 0.1, kernel: rbf
- criterion: gini, max_features: auto, n_estimators: 500
- leaf_size: 1, n_neighbors: 1

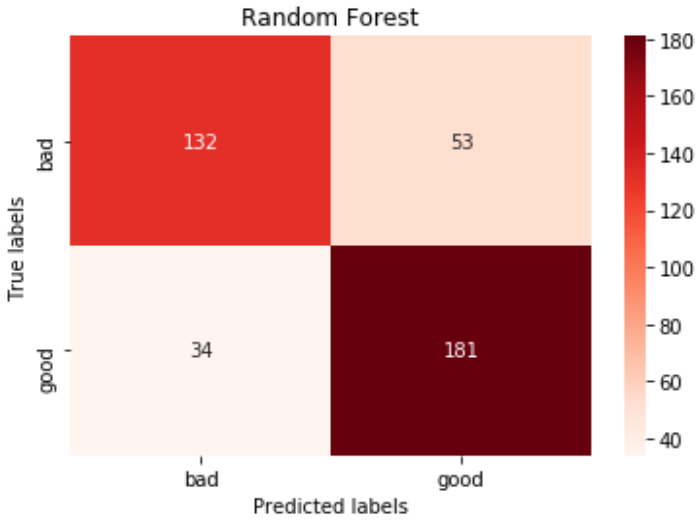
## 4. Risultati

Support Vector machines:



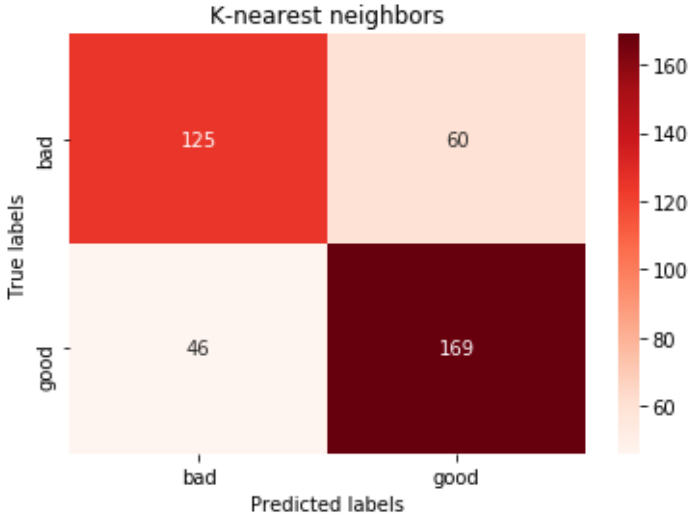

| | precision | recall | f1-score |
|---|---|---|---|
| **bad** | 0.72 | 0.71 | 0.71 |
| **good** | 0.75 | 0.76 | 0.76 |
| **accuracy** | 0.74 | | |
| **Macro avg** | 0.74 | 0.74 | 0.74 |
| **Weighted avg** | 0.74 | 0.74 | 0.74 |

**Random Forest:**



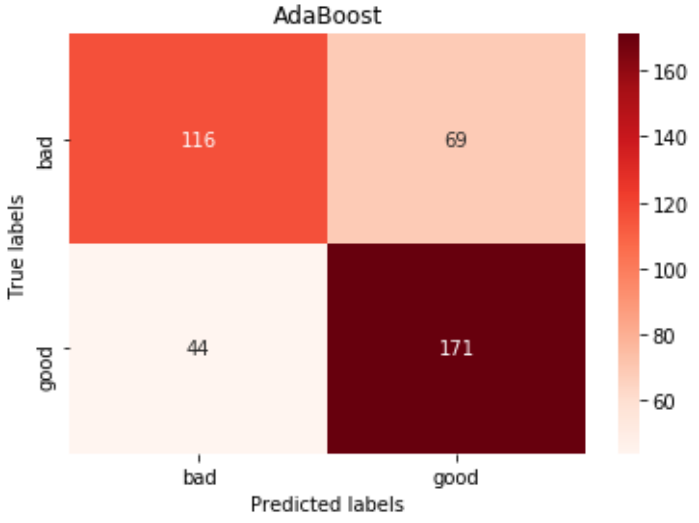

| | precision | recall | f1-score |
|---|---|---|---|
| **bad** | 0.80 | **0.71** | 0.75 |
| **good** | 0.77 | **0.84** | 0.81 |
| **accuracy** | **0.78** | | |
| **Macro avg** | 0.78 | 0.78 | 0.78 |
| **Weighted avg** | 0.78 | 0.78 | 0.78 |

K-NN:



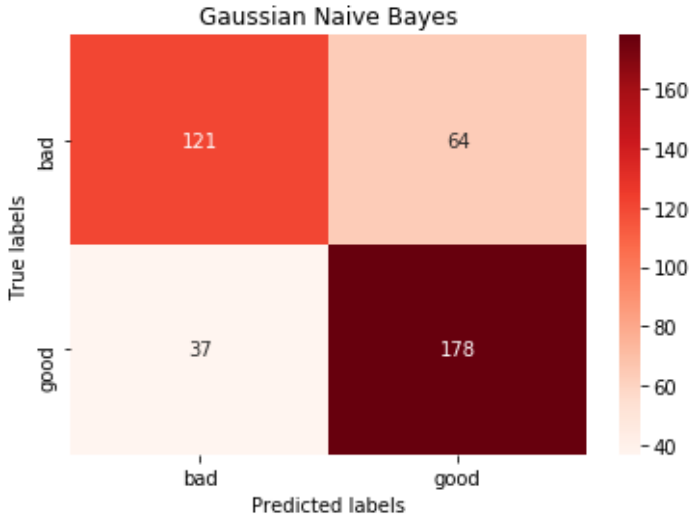

| | precision | recall | f1-score |
|---|---|---|---|
| **bad** | 0.73 | 0.68 | 0.70 |
| **good** | 0.74 | 0.79 | 0.76 |
| **accuracy** | 0.73 | | |
| **Macro avg** | 0.73 | 0.73 | 0.73 |
| **Weighted avg** | 0.73 | 0.73 | 0.73 |

Adaboost:




| | precision | recall | f1-score |
|---|---|---|---|
| **bad** | 0.72 | 0.63 | 0.67 |
| **good** | 0.71 | 0.80 | 0.75 |
| **accuracy** | 0.72 | | |
| **Macro avg** | 0.72 | 0.71 | 0.71 |
| **Weighted avg** | 0.72 | 0.72 | 0.72 |

GaussianNB:




| | precision | recall | f1-score |
|---|---|---|---|
| **bad** | 0.77 | 0.65 | 0.71 |
| **good** | 0.74 | 0.83 | 0.78 |
| **accuracy** | 0.75 | | |
| **Macro avg** | 0.75 | 0.74 | 0.74 |
| **Weighted avg** | 0.75 | 0.75 | 0.75 |

| **Algoritmo** | **Accuracy** |
|---|---|
| SVM | 0.74 |
| Random forest | 0.78 |
| K-NN | 0.73 |
| Adaboost | 0.72 |
| Naïve Bayes | 0.75 |

Daniela Grassi: d.grassi9@studenti.uniba.it, daniela.grassi.98@gmail.com

Link al progetto: https://github.com/DanielaPaw/Red-wine-quality-classifier-