Features Extraction & Red-Wine Quality Predictions

Università della Calabria

Dipartimento di Ingegneria Informatica , Modellistica , Elettronica e Sistemistica (DIMES)

Nicola Corea

Abstract. Obiettivo del seguente lavoro è quello di fornire una soluzione efficiente al problema della classificazione della qualità dei vini. Molti articoli e lavori forniscono soluzioni singole , o ensemble , garantendo prestazioni al di sotto del 70% su dati mai visti. Tramite opportune preelaborazioni dei dati , vedremo come sia possibile ottenere , soluzioni con prestazioni superiori del 20-30% di quelle esistenti. L'applicazione di tecniche di riduzione della dimensionalità del campione (selezione , estrazione delle caratteristiche) ci consentirà , a scapito di una leggerissima perdita in prestazione , di ottenere modelli più semplici e meno costosi da un punto di vista computazionale. Questo ci consentirà l'addestramento efficiente di ensemble di classificatori. L'articolo si concluderà con l'addestramento di una rete neurale multi-layer (MLP) per scopi di classificazione ed, ovviamente , ad un confronto tra le prestazioni ottenute con quelle di tecniche ensemble.

I. Intoduzione

Il seguente articolo punta alla definizione di classificatori con l'obiettivo di predire la qualità del vino sulla base delle sue caratteristiche. Per addestrare i nostri modelli faremo riferimento al dataset reperibile dal repository UCI. Il dataset , grazie al lavoro di P. Cortez , A. Cerdeira , F. Almedia , contiene 1599 varianti del vino portoghese 'Vinho Verde' classificate in base a diverse caratteristiche del vino stesso , quali ad esempio: 'fixed acidity' , 'volatile acidity' , 'citric acid' etc… . Sulla base di tali caratteristiche la qualità del vino viene rappresentata da un intero variante sull'intervallo [3,8]. L'intero '3' sarà a riferimento di una bassa qualità del vino , mentre '8' , sarà rappresentativo di una eccellenza. Per quanto riguarda la valutazione delle prestazioni del classificatore , si farà riferimento alla accuratezza del classificatore stesso. Ricordiamo brevemente , con riferimento alla matrice di confusione riportata in **Figura 1.1** , detto

$$ERR = \frac{FP + FN}{FP + FN + TP + TN}$$

l'errore nelle predizioni , l'accuratezza (ACC) del classificatore sarà allora data da

$$ACC = 1 - ERR = \frac{TP + TN}{FP + FN + TP + TN}$$

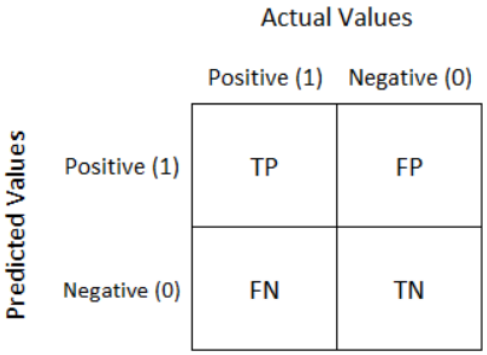

Figura 1.1 Matrice di Confusione

Inizieremo la nostra discussione sul problema proposto , partendo da una piccola fase di preelaborazione dei dati , con l'obiettivo di addestrare in maniera efficiente diversi classificatori (singoli) per la risoluzione (predizione) del problema proposto. Vedremo in accordo a [1] come le prestazioni dei singoli classificatori vanno non oltre il 70%. L'obiettivo sarà allora quello di cercare delle possibili soluzioni con l'intenzione di migliore l'accuratezza del predittore del $20 - 30$ %. Fatto ciò con l'intenzione di utilizzare tecniche ensemble , presenteremo e utilizzeremo rispettivamente tecniche di selezione e riduzione delle caratteristiche , per la riduzione della dimensionalità (caratteristiche) cercando di perdere il meno possibile in prestazioni. La trattazione di concluderà con l'addestramento di

un 'Multi – Layer Perceptron' (MLP) per scopi di classificazione. Vediamo cioè quanto una rete neurale multilivello riesce a ottenere predizioni superiori alle tecniche ensemble.

II. Analisi Del Problema

Con lo scopo di valutare le prestazioni del modello ottenuto prima di lasciarlo libero di operare nel mondo reale, suddividiamo il dataset in un 70% per l'addestramento del modello ed il restante 30% per scopi di validazione. Fatto ciò il passo successivo , e di notevole importanza è la riduzione in scala delle caratteristiche. Alcuni algoritmi di apprendimento , utilizzati nel proseguio , quale ad esempio un KNN (K-Nearest-Neighbors) o SVM (Support – Vector -Machine) a differenza di un Albero Decisionale , non è invariante rispetto alla scala delle caratteristiche. Si ricordi a tal proposito che l'individuazione dei k – vicini si traduce nella individuazione dei primi k esempi a distanza minima dal dato da classificare. Se scegliessimo come metrica la norma 2 , o norma Euclidea

$$\left|\left|x - x_k\right|\right|_2 = \sum_{i=1}^{n} (x_i - x_{ki})^2$$

appare evidente come le distanze saranno maggiormente influenzate dalle caratteristiche più elevate. Nella **Tabella 1.1** si riporta la dispersione sulla retta reale dei valori assunti da alcune delle caratteristiche. Si nota appunto , come i valori assunti da alcuni parametri quali : 'Fixed Acidity' , 'Residual Sugar' , sono notevolmente superiori rispetto alle altre. Dalle considerazioni fatte in precedenza risulta evidente la necessità di una prima fase di riduzione in scala delle caratteristiche. In particolare faremo uso della Standardizzazione. Determinata media e varianza per ciascuna caratteristica , le singole realizzazioni verranno trasformate in accordo alla seguente procedura

$$x_{std}^{i} = \frac{x^i - \mu_x}{\sigma_x}$$

, intepretabili cioè come realizzazioni di una semplice normale standard. Finita questa prima fase di preelaborazione possiamo passare al compito successivo , quello di valutare le prestazioni di alcuni fra i tanti classificatori.

	"fixed acidity"	"volatile acidity"	"citric acid"	"residual sugar"
count	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806
std	1.741096	0.179060	0.194801	1.409928
min	4.600000	0.120000	0.000000	0.900000
25%	7.100000	0.390000	0.090000	1.900000
50%	7.900000	0.520000	0.260000	2.200000
75%	9.200000	0.640000	0.420000	2.600000
max	15.900000	1.580000	1.000000	15.500000

Tabella 1.1 Dispersione Caratteristiche

possano essere facilmente estesi a situazioni multi-classe tramite la tecnica OvR. Procederemo con l'addestrare e valutare le prestazioni dei seguenti classificatori

1) KNN (K-Nearest-Neighbors)
2) SVM Kernel , con funzione kernel RBF
3) DecisionTreeClassifier , criterio Entropia

Si ricordi che come criterio di valutazione per i nostri classificatori abbiamo scelto l'accuratezza. Nella tabella qui di sotto si riportano i risultati ottenuti sia in fase di addestramento che in fase di convalida.

Classificatore	Addestramento	Test
SVM	**0.94**	**0.69**
KNN	**0.66**	**0.58**
DTC	**0.67**	**0.59**

Tabella 1.2 Risultati Singoli Classificatori

Come possiamo notare dalla tabella risultate , e come confermato in [1] , il miglior classificatore per questo tipo di problema sono di certo le macchine a vettori di supporto (SVM). Possiamo in prima battuta provare a migliorare le prestazioni attraverso tecniche ensemble. Tuttavia non avendo ancora fatto una riduzione della dimensionalità (PCA etc..) , il numero dei classificatori , per via dello elevato numero delle caratteristiche sarà limitato a 25. Si riportano di seguito in formato tabellare i risultati ottenuti con tecniche ensemble , quali: 'Majority Voting (MV)' , 'RandomForest (RF)' , 'AdaBoosting (AB)'

Ensemble	Addestramento	Test
MV	**0.79**	**0.63**
RF	**0.99**	**0.68**
AB	**0.56**	**0.54**

Tabella 1.3 Ensemble Classificatori

Per quanto riguarda il 'Majority Voting' , è stata semplicemente implementata una votazione a

maggioranza tra i precedenti singoli classificatori. Come possiamo ben osservare dalla precedente tabella , anche sfruttando tecniche ensemble , non si riescono ad ottenere buone prestazioni; l'applicazione di una RandomForest (Bagging con DecisionTree) di fatto non migliore le prestazioni rispetto al singolo SVM.

Il motivo della scarsa qualità nella fase di generalizzazione (ma anche in fase di addestramento) potrebbe di fatto essere dovuta nella non uniforme distribuzione dei campioni per classe. Come infatti riporta la **Figura 1.2** , i dati risultano maggiormente distribuiti sulle classi [5,6].Il problema potrebbe essere nel fatto che molti esempi di addestramento associati a classi differenti , hanno caratteristiche rilevanti comuni.

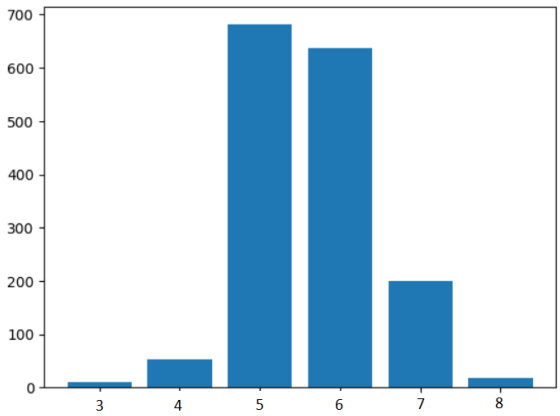

Figura 1.2 Distribuzione Esempi Per Classe

Vedremo nella prossima sezione come migliorare le prestazioni dei nostri classificatori. L'idea alla base della soluzione proposta , con l'obiettivo di massimizzare l'accuratezza , consiste , di fatto , in una seconda fase di preelaborazione dei dati , in cui si vanno a ridefinire le etichette delle classi.

III. Soluzione Proposta

Come già discusso nella precedente sezione , l'idea alla base della soluzione proposta , sta in una ridefinizione delle etichette delle classi. In particolare dalle etichette [3,4,5,6,7,8] , si passa ai label [0,1,2] corrispondenti a tre diversi livelli di qualità del vino , in particolare , saranno così definite

Label	Wine Quality
0	Low
1	Medium
2	High

Tabella 1.3 Associazione Label Qualità

Rimane , ovviamente , da stabilire come ridefinire l'etichetta della classe per ogni esempio del dataset. La trasformazione della etichetta per ciascun esempio viene fatta in accordo alla strategia presentata nella **Figura 1.3.**

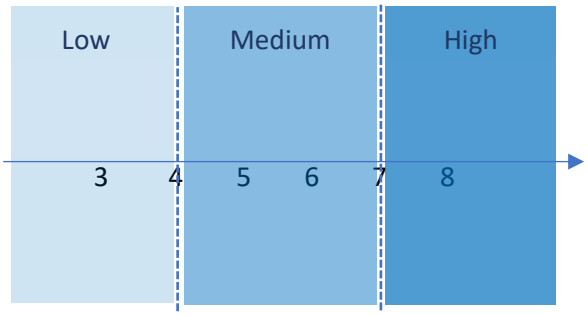

Figura 1.3 Ridefinizione Label Classi

Quindi fino a 4 la qualità del vino sarà considerata bassa . Da 5 fino a 7 la qualità sarà considera media , superiore a 7 la qualità del vino sarà giudicata alta.

Andiamo quindi a riaddestrare i nostri singoli classificatori in base a questa ridefinizione dei label delle classi , e andiamo infine a valutarne le prestazioni.

Classificatore	Addestramento	Test
SVM	0.99	0.95
DTC	0.96	0.94
KNN	0.95	0.95

Tabella 1.4 Risultati Classificatori

Notiamo dunque con questa ridefinizione delle classi del nostro dataset un aumento sulla accuratezza del 20-30%. Possiamo considerarci soddisfatti dei risultati ottenuti. Ancora una volta i classificatori SVM sono quelli che danno i

migliori risultati sia in fase di addestramento che sui dati di convalida.
I classificatori precedenti elencati , sono stati ottenuti considerando tutte e 11 le caratteristiche del dataset originale. Quello che vogliamo fare a questo punto è cercare di ridurre il numero delle caratteristiche con l'obiettivo di ottenere modelli più semplici , e quindi diminuire il costo computazionale. A tal proposito faremo uso di una nota tecnica , la PCA (Principal Component Analysis):

ALGORITMO PCA

Input : X , k
Output: W
1: Determina Matrice Covarianze Caratteristiche C
2: Determina Autovalori Autovettori di C
3: Seleziona i Primi k Autovettori Corrispondenti ai primi k Autovalori più grandi
4: Costruisci W tramite gli Autovettori Ottenuti

Prima di applicare l'algoritmo dobbiamo decidere a quante componenti fare riferimento. A tal proposito andiamo a tracciare quelli che sono i rapporti di varianza spiegata , definiti

$$\frac{\lambda_i}{\sum_{i=1}^{n} \lambda_i}$$

con λ_i allo i-esimo autovalore delle matrice delle covarianze C , e le corrispettive somme cumulative

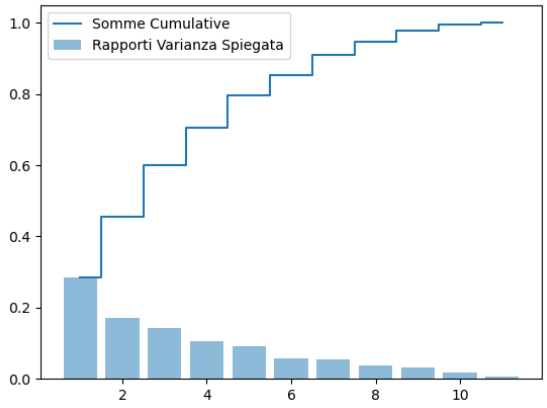

Figura 1.4 Analisi PCA

Come notiamo dalla **Figura 1.4** selezionando le prime 4 componenti abbiamo oltre il 70% percento dell'informazione portata dai dati. Quindi di fatto , scegliamo k = 4 , cioè , passiamo da uno spazio a 11 dimensione ad uno a 4 dimensioni. Si riporta di seguito un confronto tra le prestazioni prima e dopo PCA.

Siccome il miglior classificatore per il problema posto è di fatto lo SVM , l'analisi procede prendendo in considerazione solo tale predittore.

Algoritmo	Addestramento	Test
SVM	PCA	0.96	0.95
SVM		0.99	0.95

Tabella 1.5 Risultati Prima e Dopo PCA

Come possiamo osservare nella tabella precedente , abbiamo soltanto una leggera perdita in fase di addestramento sul classificatore , ma nessuna modifica per quanto riguarda la validazione del modello sui dati di test. Proviamo adesso ad applicare una ben nota tecnica di selezione delle caratteristiche , la 'Sequential Backward Selection' (SBS)

ALGORITMO SBS

Input : X , k , estimator
Output : I
1: Sia c , il numero delle caratteristiche di partenza ed I l'insieme di tutte le caratteristiche iniziali
2: **If** c > k **then:**
3: Seleziona l'insieme delle c-1 caratteristiche **p** di I che produce il miglio risultato (accuratezza)
4: Memorizza **p** e il risultato ottenuto **score**
5: Poni c = c -1 , GOTO 2
6: **else:** STOP

Applichiamo il precedente algoritmo con $k = 6$ e come stimatore sempre la macchina a vettori di supporto. Si riportano di seguito sia i risultati in fase di addestramento che di test.

Classificatore	Addestramento	Test
SVM	SBS	0.96	0.95
SVM		0.99	0.95

Tabella 1.6 Risultati ottenuti con tecnica SBS

Notiamo ancora una volta come riusciamo ad ottenere risultati al quanto soddisfacenti considerando un numero di caratteristiche inferiori. Si vuole riportare di seguito l'insieme delle caratteristiche 'ottime' individuate dallo algoritmo precedente

```
'"fixed acidity"', '"volatile acidity"', '"chlorides"',
'"free sulfur dioxide"', '"total sulfur dioxide"', '"sulphates"'
```

La **Figura 1.5** mostra l'importanza delle

caratteristiche calcolata in base alla impurità media determinata da un insieme di classificatori ad albero decisionale

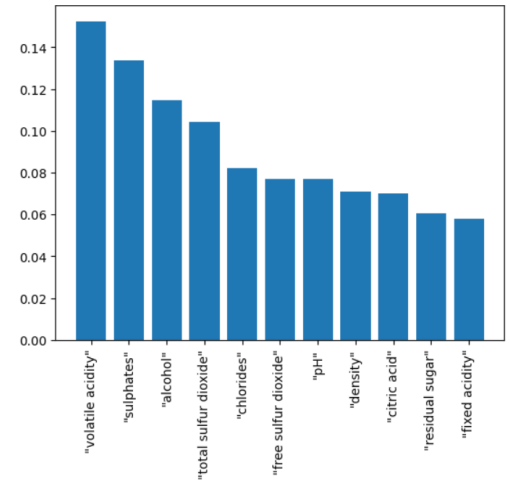

Figura 1.5 Valutazione Importanza Caratteristiche

È interessante notare come le caratteristiche rilevate dall' algoritmo SBS rientrano le caratteristiche che hanno più significative come specificato nella figura sopra riportata.

Concludiamo la trattazione al problema posto , mostrando i risultati ottenuti da una rete neurale multi-layer (MLP – MultiLayer Perceptron) per scopi di classificazione.

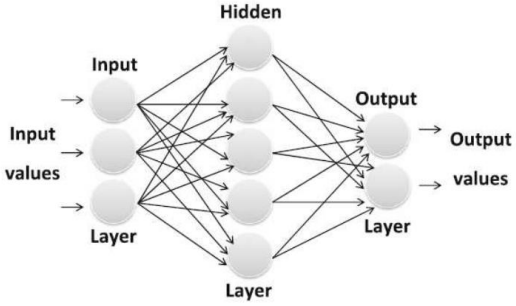

Figura 1.6 Multi-Layer Perceptron

Per quanto riguarda la scelta dei layer , addestreremo una rete neruale con un solo layer nascosto , costituito da 8 neuroni. In accordo ad una codifica one-hot delle classi , il layer di output sarà costituito da 3 soli neuroni.
Per evitare il problema della sparizione dei gradienti , e quindi di evitare la possibilità di andare incontro a minimi locali , come funzione di attivazione per il layer nascosto impiegheremo la funzione ReLU ricordiamolo definita come

$$\phi(z) = \max(0, z)$$

per quanto riguarda invece la funzione di attivazione del layer di output , sceglieremo come funzione di attivazione , la funzione softmax

$$\phi(z) = \frac{e^{z_i}}{\sum_{j=1}^{t} e^{z_j}}$$

che come ricordiamo , ci da risultati significativi riguardanti le probabilità di appartenenza di un campione ad una classe.

Infine per quanto riguarda la funzione da minimizzare , il criterio di valutazione , faremo riferimento alla 'Cross-Entropy' definita

$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

Scegliendo come numero delle epoche di addestramento del nostro modello , 20 epoche , si riportano in formato i risultati ottenuti sia in fase di addestramento che di convalida del modello.

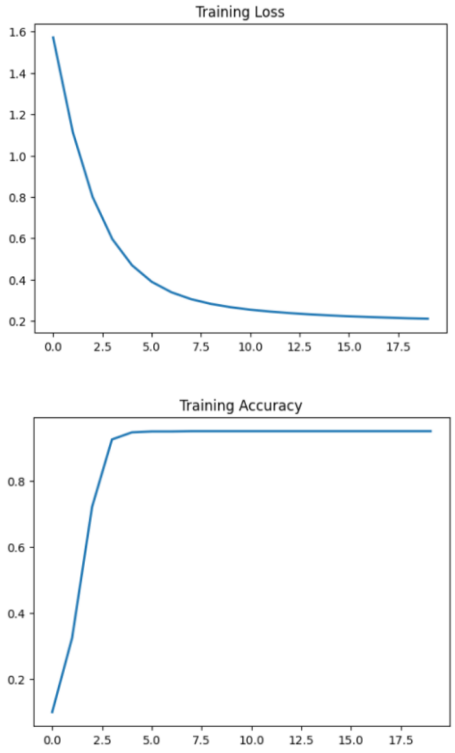

Figura 1.7 Risultati Addestramento MLP

Loss A.	Acc A.	Loss T.	Acc T.
0.2108	0.9491	0.2830	0.95

Tabella 1.7 Risultati Tabellati MLP

Riferimenti

[1] S. Kumar , K. Agrawal , N. Mandan , "Red Wine Quality Prediction Using Machine Learning Techniques", (ICCCI -2020)

[2] K. R. Dahal , J. N. Dahal , H. Banjade , S. Gaire "Prediction of Wine Quality Using Machine Learning Algorithms" , Open Journal of Statistics, 2021, 11, 278-289

[3] Y. Gupta "Selection of Important Features and Predicting Wine Quality using machine learning techniques" , ICSCC -2017.

[4] S. Rashka "Python Machine Learning" , Book-2015.

[5] S. Mustafa "Feature Selection Using Backward Method in Melanoma Recognition" , UTC from IEEE Xplore - 2023.