Relazione progetto Machine Learning

Appello Febbraio 2021

Gabriele Maria Bucosse 869940
Ruggero Panzeri 874975
Davide Rendina 830730

19, February 2021

Contents

1 Introduzione

L'obiettivo di questo elaborato è la presentazione del progetto svolto di *machine learning*, relativo alla classificazione di differenti varietà di vino. Nella relazione sono incluse le fasi di esplorazione, generazione dei modelli, esperimenti e calcolo delle performance.
Il dataset scelto per lo svolgimento del progetto è *Red Wine Quality* [Cortez et al., 2009], disponibile sulla piattaforma *UCI Machine Learning Repository*. Il dataset è relativo alle varianti del vino portoghese *Vinho Verde* e include 11 covariate di input, che presentano valori basati su alcune rilevazioni fisico-chimiche, e 1 di output, il cui valore è ottenuto stabilito combinando i dati di sensoristica e almeno 3 valutazioni fatte da esperti vinicoli. Le singole *feature* sono approfondite nella successiva tabella (1).

Per lo svolgimento del progetto, sono state svolte diverse attività. La prima fase ha riguardato l' esplorazione del *dataset* attraverso alcuni grafici, analisi di covariate e informazioni ottenute dalla *PCA*. Successivamente, sono stati addestrati e testati 6 differenti modelli per effettuare la classificazione dei vini. In base ai risultati ottenuti sono state calcolate diverse metriche e, analizzando i valori ottenuti, è stato infine possibile effettuare un confronto riguardante le performance dei vari modelli.

Feature	**Descrizione**
Fixed acidity	La maggior parte degli acidi del vino, fissi o non volatili (che non evaporano facilmente)
Volatile acidity	La quantità di acido acetico nel vino, che a livelli troppo alti può portare a un sapore sgradevole di aceto
Citric acid	Presente in piccole quantità, l'acido citrico può aggiungere "freschezza" e sapore ai vini
Residual sugar	La quantità di zucchero rimanente dopo lo stop della fermentazione, è raro trovare vini con meno di 1 grammo/litro e i vini con più di 45 grammi/litro sono considerati dolci
Chlorides	La quantità di sale nel vino
Free sulfur dioxide	La forma libera di SO2 esiste in equilibrio tra SO2 molecolare (come gas disciolto) e ione bisolfito; previene la crescita microbica e l'ossidazione del vino
Total sulfur dioxide	Quantità di forme libere e vincolate di S02; a basse concentrazioni, SO2 è per lo più non rilevabile nel vino, ma a concentrazioni superiori a 50 ppm, SO2 diventa evidente all'odore e al gusto del vino
Density	La densità dell'acqua è influenzata dalla percentuale di alcol e dalla quantità 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	Additivo per vino che può contribuire ai livelli di anidride solforosa (S02), che agisce come un antimicrobico e antiossidante
Alcohol	Percentuale di alcool contenuta nel vino
Quality	Rappresenta la **variabile target** del dataset e indica la qualità del vino, basata sui dati delle feature presentate in precedenza. Il punteggio è rappresentato da un valore nel range 1-10.

Table 1: Descrizione delle feature presenti nel dataset

2 Analisi del dataset

In questo capitolo viene approfondita la fase di analisi del dataset, organizzata in 2 differenti sezioni che descrivono rispettivamente le attività svolte di esplorazione iniziale e *PCA*.

Obiettivo della fase iniziale di esplorazione è il controllo riguardante la correttezza del *dataset* caricato e l'analisi delle covariate e le relazioni che intercorrono tra esse. La *PCA* invece, è stata svolta per effettuare un ulteriore studio sulle relazioni tra le covariate ma soprattutto al fine di ridurre la dimensionalità dei dati e contestualmente il loro rumore.

2.1 Esplorazione del dataset

Inizialmente sono state visualizzate le prime righe del *dataset* (figura 1), per effettuare una prima esplorazione delle variabili e valori presenti. Come è possibile osservare dal grafico, i vini sono descritti solamente da *feature* numeriche. Si può notare inoltre che il *dataframe* presenta esattamente 12 covariate, numero che coincide con quanto indicato nella descrizione del *dataset* [1]. Un ulteriore controllo sulla dimensionalità dei dati ha fornito risultati relativi a numero di colonne (12) e istanze caricate (1599) che hanno trovato anch'essi corrispondenza con quanto dichiarato nella descrizione del dataset.

	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	density	pH	sulphates	alcohol	quality
1	7.4	0.70	0.00	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	0
2	7.8	0.88	0.00	2.6	0.098	25	67	0.9968	3.20	0.68	9.8	0
3	7.8	0.76	0.04	2.3	0.092	15	54	0.9970	3.26	0.65	9.8	0
4	11.2	0.28	0.56	1.9	0.075	17	60	0.9980	3.16	0.58	9.8	1
5	7.4	0.70	0.00	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	0
6	7.4	0.66	0.00	1.8	0.075	13	40	0.9978	3.51	0.56	9.4	0
7	7.9	0.60	0.06	1.6	0.069	15	59	0.9964	3.30	0.46	9.4	0
8	7.3	0.65	0.00	1.2	0.065	15	21	0.9946	3.39	0.47	10.0	2
9	7.8	0.58	0.02	2.0	0.073	9	18	0.9968	3.36	0.57	9.5	2
10	7.5	0.50	0.36	6.1	0.071	17	102	0.9978	3.35	0.80	10.5	0

Figure 1: Prime righe del dataframe

Successivamente è stata controllata anche la presenza di valori *NA*, che ha dato esito negativo (0 *NA* riscontrati) come già indicato nella descrizione del *dataset*. Questo risultato indica che non è necessaria nessuna politica per la gestione di valori mancanti. Infine è stato effettuato un controllo per analizzare la struttura del *dataset* e le tipologie di dati presenti, che ha evidenziato come tutte le covariate fossero di tipo numerico (sia discrete che continue) (figura 2).

I risultati ottenuti da questa esplorazione preliminare ci permettono quindi di affermate che durante il caricamento del dataset non è presente alcun errore, in quanto tutti i valori ottenuti hanno avuto riscontro positivo con quanto dichiarato nella panoramica del dataset.

Al termine dei controlli riguardanti la correttezza dei dati caricati è stata scelta la covariata *quality* come variabile target, poiché considerata come la più significativa per la predizione (oltre ad essere indicata come tale nella descrizione del dataset). Successivamente è stata controllata attraverso un *barplot* la sua distribuzione dei valori (figura 3), dal quale emerge che le istanze siano fortemente sbilanciate. Infatti, la maggior parte dei vini che ha qualità tra i valori 5-6 (82% delle istanze) e una minoranza che invece presenta qualità bassa (4%) oppure alta (14%). Nel dataset inoltre non è presente alcuna istanza

[1] UCI Machine Learning: Wine Quality

```
fixed acidity     volatile acidity          citric acid       residual sugar
    "numeric"            "numeric"            "numeric"            "numeric"
    chlorides  free sulfur dioxide total sulfur dioxide              density
    "numeric"            "numeric"            "numeric"            "numeric"
           pH            sulphates              alcohol              quality
    "numeric"            "numeric"            "numeric"            "numeric"
```

Figure 2: Tipi di valori presenti nelle covariate

con valori 1-2 oppure 9-10. La distribuzione della variabile riflette la distribuzione dei vini nel mondo reale, che vede quelli di qualità media in numero significativamente maggiore rispetto a quello di cattiva e buona qualità.

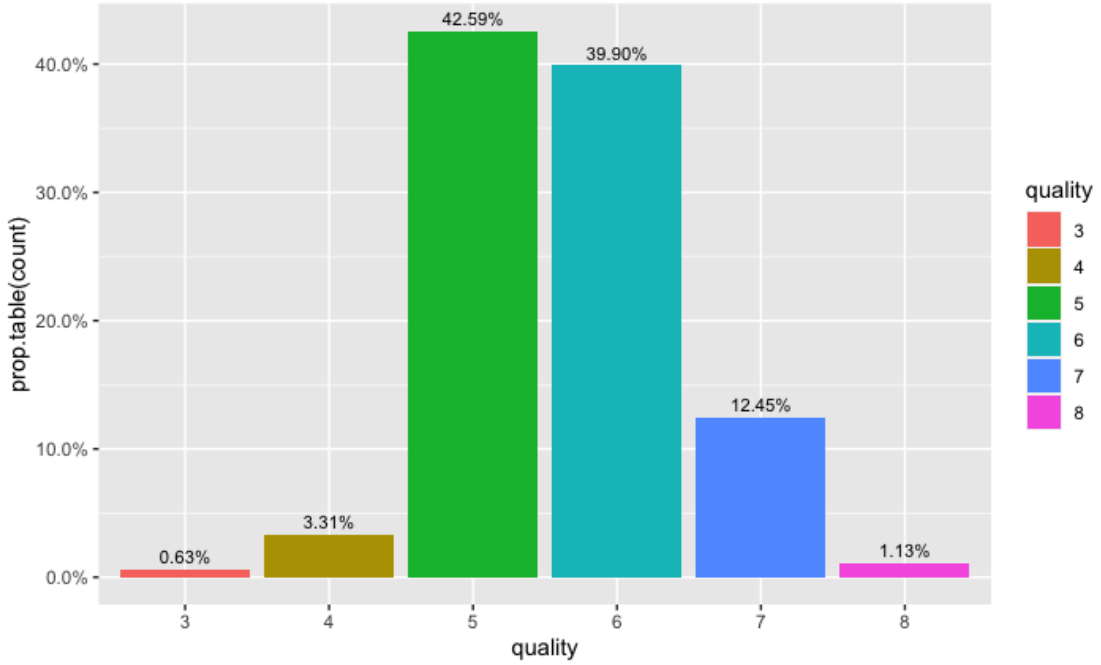

Figure 3: Distribuzione dei valori della feature quality

Dopo aver riscontrato questa distribuzione delle classi abbiamo scelto di ridurre il loro numero, al fine di facilitare i modelli di riconoscimento nella loro classificazione. Per effettuare questa suddivisione, sono state valutate diverse strategie:

1. Ridurre le classi totali da 10 a 5, accoppiando quindi le classi vicine (1-2, 3-4, 5-6, 7-8, 9-10). Il risultato sarebbero quindi state solamente 3 classi (siccome 1-2 e 9-10 non sono presenti), ma le istanze risulterebbero comunque fortemente sbilanciate (14%, 82% e 4% - in ordine di qualità)

2. Trasformare il problema a un problema binario, considerando le classi vini insufficienti (qualità minore di 6) e sufficienti (qualità maggiore o uguale a 6), ottenendo così due classi bilanciate (47% e 53% rispettivamente).

3. Focalizzandoci solamente sui valori a disposizione, è possibile identificare 3 diverse classi possibili: 3-5 (qualità bassa oppure *LOW*), 6 (qualità media oppure *MEDIUM*) e 7-8 (qualità alta oppure *HIGH*). Le classi risultano comunque sbilanciate (rispettivamente 46.5%, 39.9% e 13.6%), ma la situazione è migliore rispetto a quanto analizzato nel punto 1.

Si può notare quindi come la soluzione ottimale sia quella presentata nel punto 2, con un problema binario e un bilanciamento quasi perfetto tra le classi. La nostra scelta però è ricaduta sulla terza opzione, poiché:

- Consideravamo la suddivisione binaria un po' meno significativa, dato che la divisione da noi scelta mantiene concettualmente un livello di qualità in più per la catalogazione dei vini.
- Ci piaceva l'idea di confrontarci con un problema multiclasse e non solo binario.

Dopo aver scelto la suddivisione, è stata impostata a categorica la covariata *quality*. La nuova distribuzione di questa feature è illustrata nel *barplot* in figura 4, che fornisce anche informazioni sulla buona riuscita dell'operazione svolta. É possibile infatti verificare che i livelli di *quality* sono 3 e non più 6 e che le percentuali cumulate corrispondono. Un ulteriore controllo sulle righe del *dataframe* ha constatato che è stato modificato il numero delle istanze.

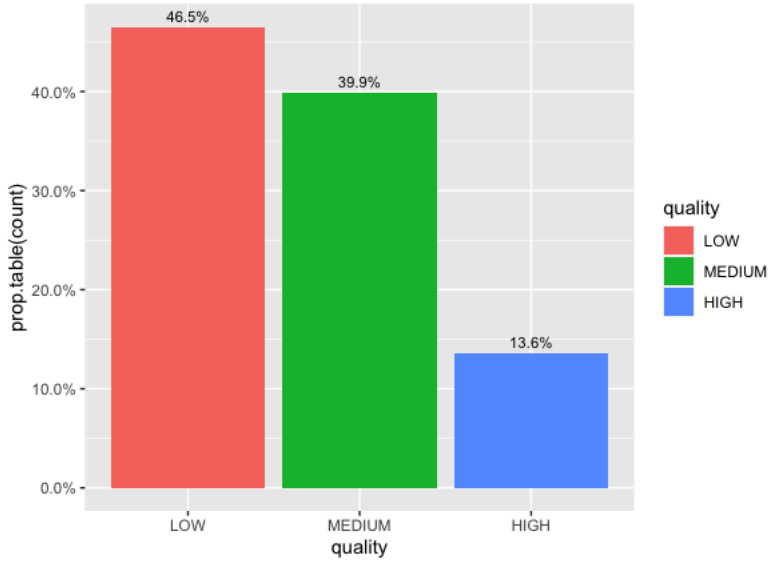

Figure 4: Nuova distribuzione dei valori della feature quality

Sono stati visualizzati successivamente diversi grafici per l'analisi delle covariate del *dataset*, al fine di comprendere la difficoltà del problema e le studiare relazioni tra le varie *feature*.

Per ogni covariata è stato generato un *boxplot* che mette in relazione i suoi valori con quelli della variabile target. Da questi grafici evince come in realtà nessuna tra le *feature*

del *dataset* sia in grado di effettuare una distinzione efficace tra le classi, poiché per tutte le istanze si verifica la sovrapposizione non solo dei "baffi" del grafico, ma anche dei relativi box (dove si concentra la maggior parte dei valori). Inoltre per alcune covariate sono presenti molti *outliers*, che potrebbero causare problemi nella fase successiva di classificazione. Dai grafici si può notare come le covariate *residual.sugar* e *chlorides* siano in assoluto le peggiori, in quanto hanno quasi completamente i box sovrapposti e numerosi *outliers*. Un'altra covariata che presenta un numero notevole di *outliers* è *sulphates*. Le *feature* migliori per la discriminazione (anche se comunque non presentano risultati buoni) sembrano essere *alcohol* e *volatile.acidity*, in quanto le uniche per le quali i box non sono molto sovrapposti, le mediane sono abbastanza lontane per tutte e tre le classi e non presentano molti *outliers*.

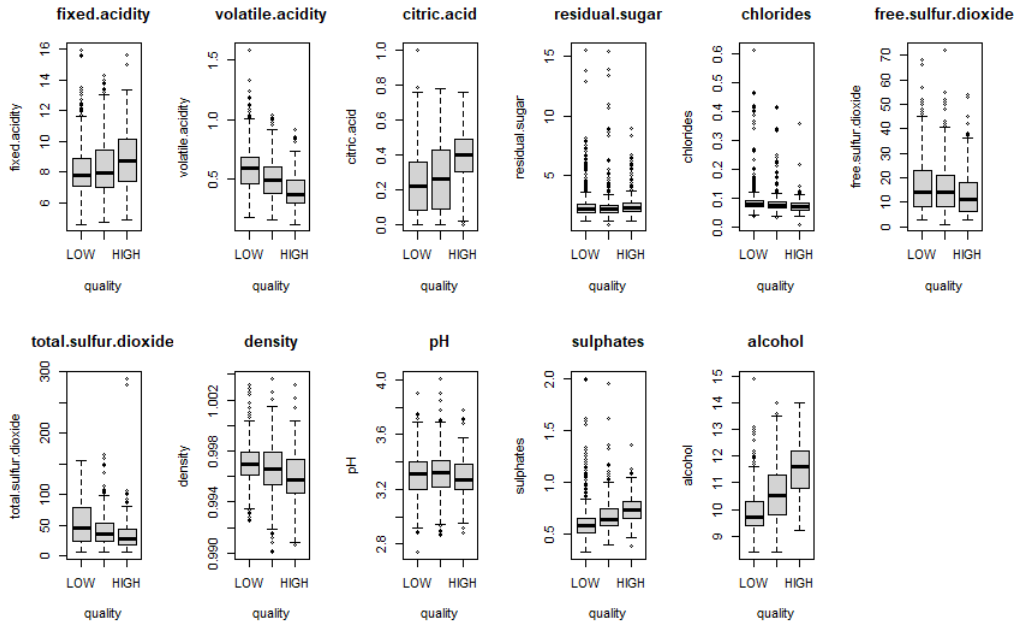

Figure 5: Output boxplot per label

Anche attraverso la visualizzazione dello *scatterplot* per tutte le covariate (figura 7), ci si può rendere conto come in realtà nessuna coppia di esse sia in grado di effettuare una divisione efficace delle classi. Le classi infatti risultano per tutte le combinazioni di molto sovrapposte, e questa condizione è indice del fatto che potrà essere complicato avere una suddivisione efficace delle classi. Le covariate che sembrano suddividere meglio in questo caso le tre classi sembrano essere *free.sulfur.dioxide* e *total.sulfur.dioxide*. Anche osservando la distribuzione delle varie *feature* (figura 6) si può osservare come i valori risultino fortemente sovrapposti e per alcune covariate sono perfettamente sovrapposte anche le distribuzioni stesse.

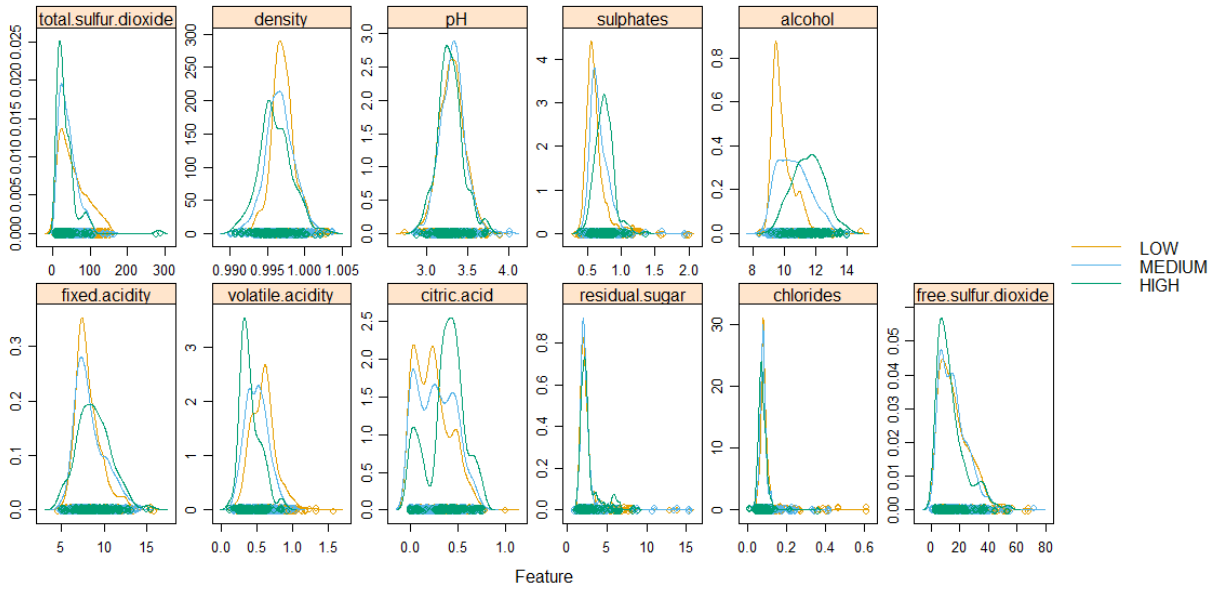

Figure 6: Distribuzione delle variabili

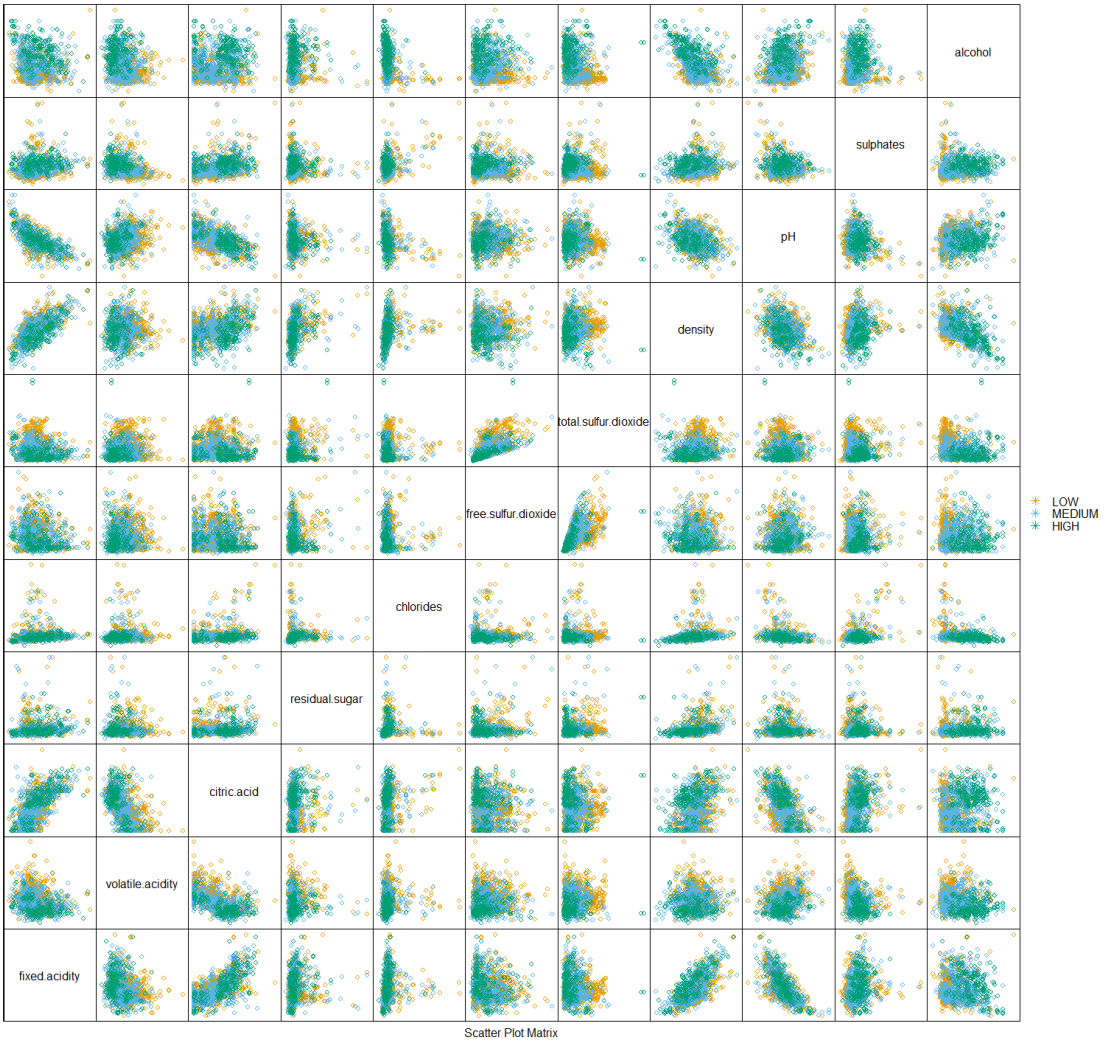

Figure 7: Scatterplot Matrix

2.2 PCA: Principal Component Analysis

Al fine di studiare in maniera più approfondita la correlazione tra le variabili e contestualmente provare a ridurre la dimensionalità del dataset e il rumore nei dati, è stata effettuata anche la *Principal Component Analysis*. In un *dataset* non tutti gli attributi sono assolutamente informativi (si può vedere anche dai vari grafici visti finora) ed alcuni possono anche risultare dannosi, aumentando solamente la complessità del modello e rischiando in alcuni casi di confonderlo. Per questi motivi, è stato deciso di utilizzare questo metodo.

Dopo aver effettuato la *PCA*, è stata osservata la varianza rappresentata da ogni componente. Come si può vedere dagli autovalori presenti nella figura 8, non sono presenti

componenti che singolarmente spiegano una grande quantità di varianza, ma piuttosto la varianza è distribuita in maniera abbastanza omogenea tra diverse le componenti (escluse ovviamente le ultime).

	eigenvalue	variance.percent	cumulative.variance.percent
Dim.1	3.09913244	28.1739313	28.17393
Dim.2	1.92590969	17.5082699	45.68220
Dim.3	1.55054349	14.0958499	59.77805
Dim.4	1.21323253	11.0293866	70.80744
Dim.5	0.95929207	8.7208370	79.52827
Dim.6	0.65960826	5.9964388	85.52471
Dim.7	0.58379122	5.3071929	90.83191
Dim.8	0.42295670	3.8450609	94.67697
Dim.9	0.34464212	3.1331102	97.81008
Dim.10	0.18133317	1.6484833	99.45856
Dim.11	0.05955831	0.5414392	100.00000

Figure 8: Autovalori PCA

Per scegliere quali componenti da mantenere abbiamo deciso di non considerare solamente quelle con autovalori con valore maggiore di 1 ma di scegliere in base alla varianza cumulata, al fine di mantenere un valore di varianza alto. In particolare, è stato deciso di tenere solamente 7 componenti in maniera tale da avere il 90% della varianza totale. Per effettuare poi le operazioni successive di *train* e *test* è stato considerato il *dataset* prodotto dalla *PCA* nel nuovo spazio di indirizzamento, passando da 11 a 7 dimensioni.

Analizzando le prime due dimensioni della PCA, è possibile effettuare qualche considerazione attraverso alcuni grafici. Osservando il grafico delle variabili (figura 9), si può notare come *sulphates, chlorides e anche residual.sugar* siano le *feature* peggio rappresentate nel nuovo spazio di indirizzamento. Riprendendo le considerazione fatte in precedenza analizzando i boxplot, si può notare come queste siano le variabili che avevamo indicato come le peggiori, sia per sovrapposizione dei valori sia per la presenza di numerosi *outliers*.

Analizzando questo grafico, si possono effettuare anche considerazioni sulla correlazione tra le variabili. Le covariate *fixed.acidity, citric.acid e sulphates* risultano fortemente dipendenti tra loro, come accade per *total.sulfur.dioxide e free.sulfur.dioxide* e anche *chlorides e density*. In generale, la maggior parte delle variabili risulta linearmente dipendente (più o meno fortemente), in quanto le frecce sono tutte concentrate nel quadrante in alto a dx (o nelle sue vicinanze). Le *feature alcohol*, *pH* e *volatile.acidity* risultano invece poco correlate rispetto a tutte le altre (soprattutto *alcohol*).

Un altro grafico sul quale si possono effettuare alcune considerazioni è quello degli *individuals* (figura 10). Si può verificare come nel nuovo spazio di rappresentazione (considerando solo le prime due dimensioni), molte istanze non vengano ben rappresentate (colorate di azzurro e verde, concentrate al centro del grafico). Inoltre dal grafico riguardante la classificazione si può osservare che le sole prime due dimensioni non siano necessarie per una suddivisione efficace delle istanze, in quanto tutti gli elementi risultano fortemente sovrapposti.

Questi risultati ci suggeriscono che anche utilizzando il nuovo spazio di indirizzamento, potremmo avere comunque problemi ad individuare le diverse classi (questo era un risul-

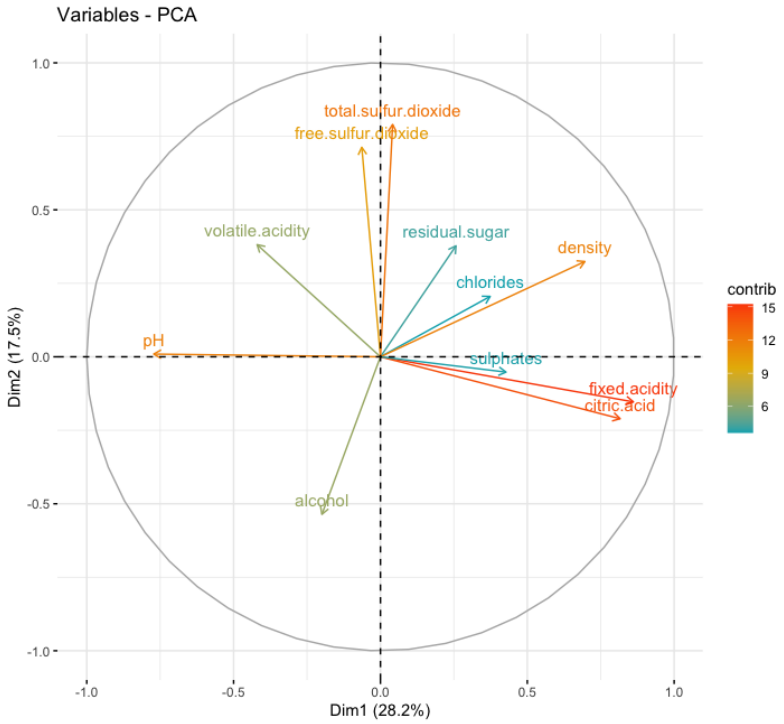

Figure 9: Output fvizpcavar()

tato atteso, in quanto abbiamo ridotto la varianza e comunque le due dimensioni spiegano solamente il 45% della varianza totale). Avendo rimosso però una parte di informazioni ridondanti, potremmo avere come beneficio un aumento della precisione dei modelli, nonché una minore complessità degli stessi.

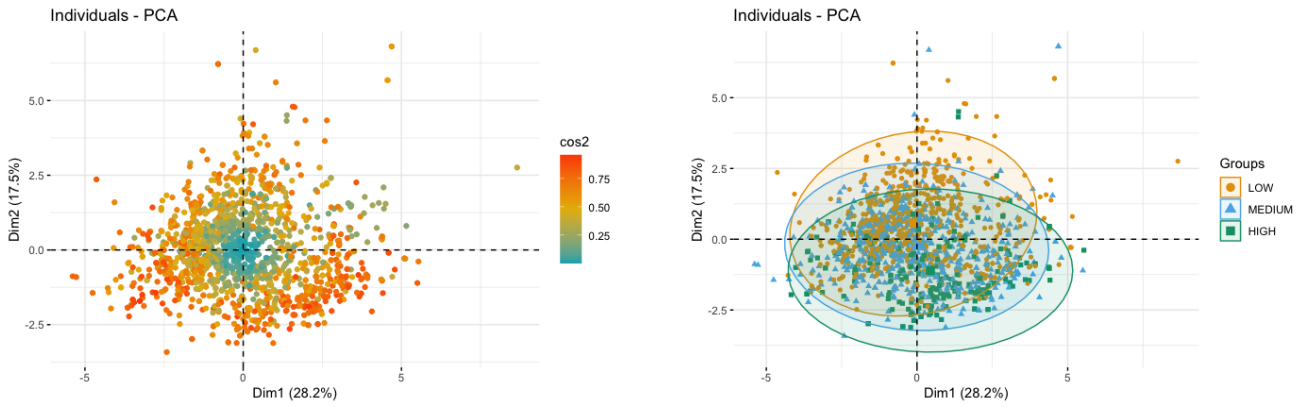

Figure 10: Individuals PCA

3 Esperimenti

Questo capitolo approfondisce la parte del progetto relativa ai modelli e agli esperimenti effettuati. In particolare vengono motivate le decisioni effettuate nella scelta e addestramento dei modelli, descritto il *workflow* seguito per ogni modello e analizzati i risultati ottenuti.

3.1 Scelta dei modelli

Per identificare i modelli più adatti, è stato scelto un approccio che prevede il confronto tra sei diversi modelli di *machine learning*, per confrontare le loro prestazioni e identificare i migliori. In particolare, abbiamo utilizzato per i nostri esperimenti una rete neurale, un albero di decisione, *naive bayes* e tre diversi modelli di *SVM* (con *kernel* rispettivamente lineare, polinomiale, radiale). Abbiamo escluso solamente l'algoritmo *k-means*, in quanto il nostro desiderio era effettuare solamente approcci supervisionati, avendo anche a disposizione le *label* per le istanze. È stato possibile scegliere questo tipo di approccio poiché il dataset utilizzato presenta un numero esiguo di istanze e *feature*, e il rispettivo *train* dei vari modelli non è molto pesante in termini di tempo e risorse utilizzate. Se invece avessimo avuto un numero notevolmente maggiore di istanze e covariate, sarebbe stato più complicato sotto certi aspetti effettuare queste operazioni per tutti quanti i modelli.

3.2 Attività svolte per ogni modello

Per ogni modello introdotto in precedenza, si è seguito il seguente *workflow*:

1. Addestramento attraverso il *trainset*
2. Predizione sui dati contenuti nel *testset*
3. Calcolo delle metriche per il modello
4. Calcolo della curva ROC multiclasse

Train del modello e predizione Per la suddivisione dei dati in *trainset* e *testset* è stato deciso di selezionare il 30% delle istanze complessive (480 istanze) in modo da rispettare la distribuzione delle istanze in base al loro valore di *quality*.

L'addestramento del modello è stato effettuato svolgendo la *3-repeated 10-fold cross validation*, attraverso il parametro *trainControl* della funzione *caret::train*. Questa opzione ci ha permesso di fare il *tuning* ottimale dei parametri dei vari modelli, in quanto effettua diverse iterazioni con valori differenti al fine di trovare la misura che permette di ottimizzare la misura specificata. Per effettuare questa ottimizzazione, è stata scelta la misura di AUC.

Le predizioni invece sono state effettuate utilizzando il modello generato dal *training* e il *testset* precedentemente suddiviso.

Calcolo delle metriche Per ogni predizione effettuata, è stata poi calcolata la matrice di confusione. Il calcolo della matrice ci ha quindi permesso di determinare le differenti metriche per ogni modello generato. In particolare è stato possibile calcolare la *accuracy* su tutta la matrice di confusione e *precision, recall e f-measure* per ogni classe del problema. I risultati di questa fase verranno discussi nella sezione successiva.

Calcolo della curva ROC É stato effettuato infine il calcolo della curva *ROC* e del relativo valore di *AUC*. In quanto problema multiclasse, sono state generate 5 diverse curve per ogni modello: 3 relative alle classi del problema (*LOW, MEDIUM, HIGH*) e le altre 2 invece alla *micro average* e *macro average* dei valori ottenuti.

3.2.1 Repeated 10-fold cross validation

Per effettuare il *train* dei modelli, sono state valutate diverse tipologie differenti di metodi di *resampling*. In particolare, sono state testate le prestazioni per *10-fold cross validation* (d'ora in poi *10-fold CV*) e *k-repeated 10-fold cross validation* (considerate 10, 5 e 3 ripetizioni).

Dopo aver testato i nostri modelli utilizzando la *10-fold CV*, è stato deciso di sperimentare anche la *10-fold CV k-repeated*. Partendo da un valore di ripetizioni k=10, abbiamo successivamente diminuito il numero di ripetizioni a k=5 e k=3. Durante lo svolgimento di queste tre prove, non sono stati riscontrati miglioramenti significativi utilizzando diversi valori di k, soprattutto non tali da giustificare l'aumento di complessità computazionale temporale.

Modello	**10-fold CV**	**3 repeated**	**5 repeated**	**10 repeated**
Albero decisionale	1.41	3.09	4.91	8.67
Naive Bayes	1.36	3.05	4.44	8.08
Svm lineare	2.96	3.33	5.03	9.23
Svm polinomiale	27.77	81.91	138.83	272.82
Svm radiale	4.86	13.08	20.97	41.13
Rete neurale	8.23	24.00	38.24	74.36

Table 2: Tempi di calcolo con diversi metodi di resampling

I grafici successivi (figure 11 e 12) mettono a confronto i valori delle metriche ottenuti attraverso la *10-fold CV* (viola) con quelli risultanti dalla *10-cross CV 3-repeated* (verde), per ogni metrica e modello. Sono presentati in particolare due grafici, uno relativo alla classe *high* e l'altro al valore di *macro average* tra tutte e tre le classi; i grafici delle altre due classi (*medium, low*) non sono stati considerati per questo confronto in quanto poco significativi.

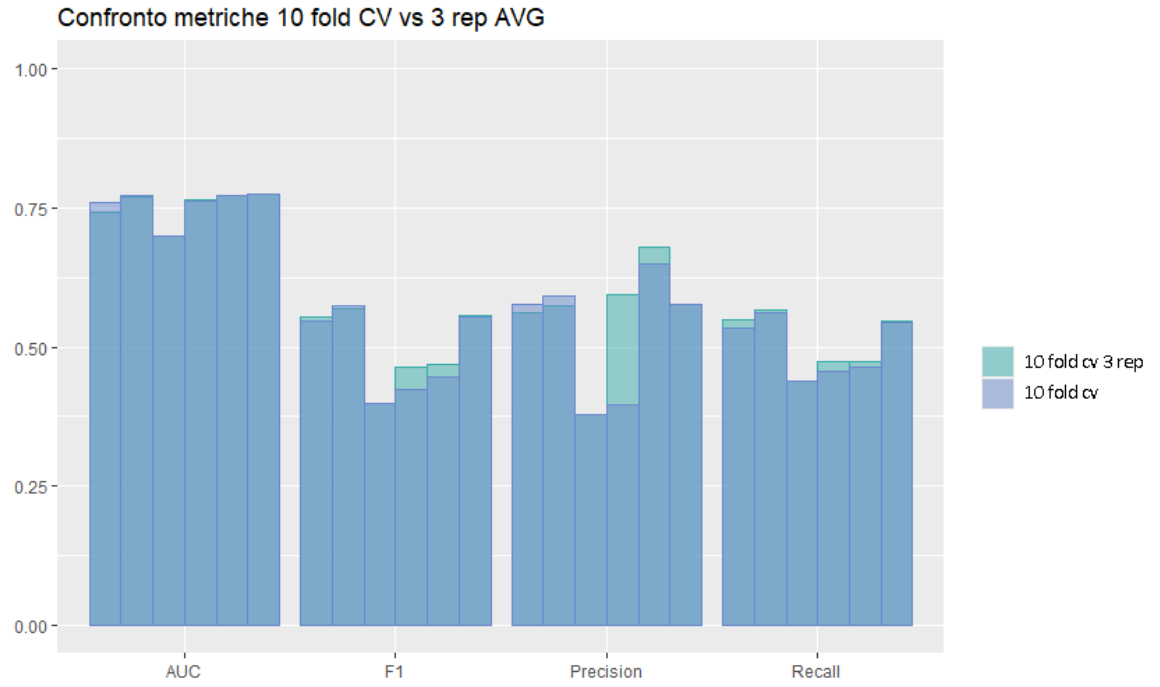

Figure 11: Confronto relativo alla media

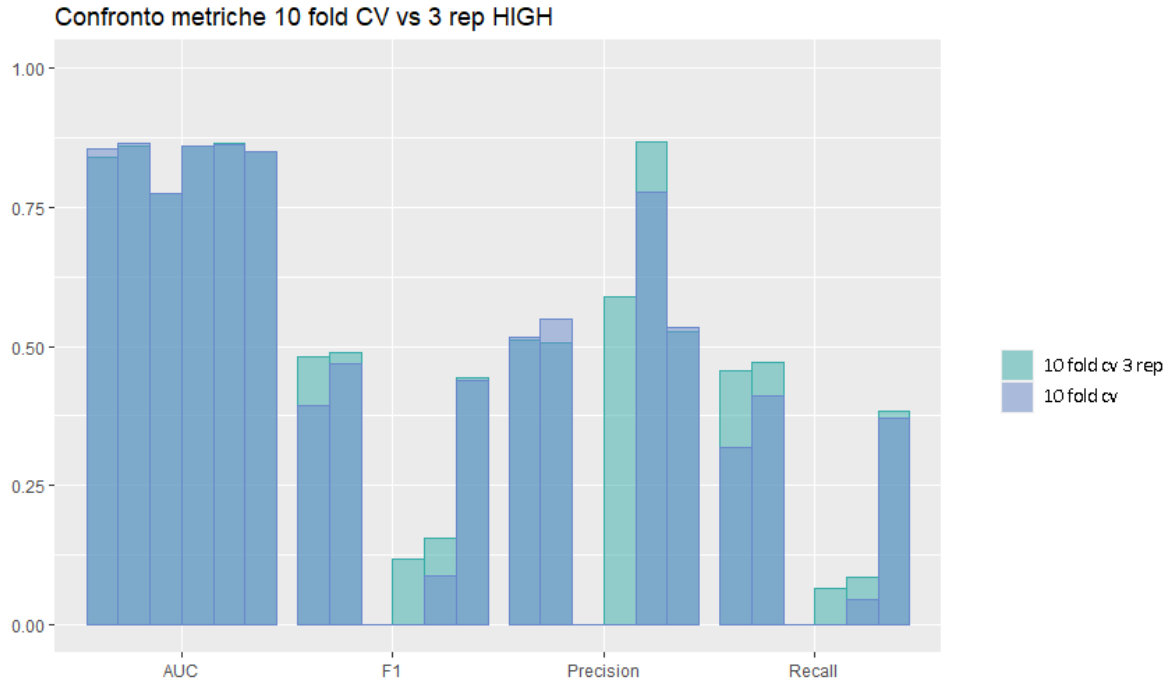

Figure 12: Confronto relativo alla classe high

Dai valori di *macro average* (11) emerge che ci sono stati piccoli cambiamenti in positivo per diversi modelli per le metriche *F1*, *recall* e soprattutto *precision*. I cambiamenti più interessanti possono essere però osservati nel grafico successivo (12), dove emerge un significativo miglioramento per tutte le metriche considerate. Uno dei modelli considerati inoltre non era in grado di classificare correttamente alcuna istanza per la classe *high*, situazione che invece è cambiata con l'utilizzo della *10-fold CV 3-repeated*. Questi motivi hanno quindi spinto all'adozione della *10-fold CV 3-repeated* a favore della *10-fold CV* senza ripetizioni.

3.2.2 Downsampling

Essendo il numero di istanze delle varie classi sbilanciato, è stato osservato che il comportamento dei modelli risultava migliore sulla classe di maggioranza (LOW) rispetto alle altre classi con meno istanze (*MEDIUM, HIGH*). Per cercare di migliorare le prestazioni per le classi meno numerose, è stato effettuato l'addestramento delle stesse applicando una strategia di *downsampling*. É stato scelto di utilizzare questa tecnica a favore di altre per due motivi principali:

- Le altre tecniche creavano un numero molto alto di istanze per le classi di minoranza e andavano ad inserire numerosi istanze che non riflettevano la realtà.
- Essendo i valori di tutte le *feature* molto vicini tra le varie classi, la creazione di altre istanze avrebbe rischiato di confondere ulteriormente il modello.

Analizzando i risultati ottenuti , abbiamo verificato che questa strategia ci ha permesso di migliorare le prestazioni in termini di riconoscimento della classe di minoranza *HIGH*, ma ha anche peggiorato notevolmente le prestazioni per la classe *MEDIUM* e anche i valori di *macro average* sono risultati leggermente inferiori. Per questi motivi, è stato deciso quindi di non adottare nessuna strategia di questo tipo.

3.3 Analisi dei risultati ottenuti

La prima metrica che è stata considerata per effettuare l'analisi è la *accuracy*, i quali valori sono riportati nel grafico 13 insieme all'intervallo di confidenza. Come si può osservare dalla figura, tutti i modelli riportano sia valori che intervalli quasi sovrapposti. Considerando però che stiamo analizzando un problema multiclasse questi valori da soli non sono sufficienti per valutare la bontà del modello, in quanto non tengono conto delle diverse classi presenti. Per effettuare quindi analisi più accurate, dobbiamo considerare anche ulteriori metriche.

Avendo scelto un problema multiclasse, abbiamo ottenuto diversi valori per le metriche calcolate. Per la visualizzazione dei valori di AUC e metriche, sono stati utilizzati grafici a barre. Sono stati generati in tutto 4 grafici di questa tipologia, 3 relativi alle classi del problema mentre il quarto prende in considerazione la *macro average*, ed oguno di essi mette in relazione per ogni misura i risultati ottenuti dai vari classificatori (figura 3.3). Abbiamo valutato questa visualizzazione come la migliore poiché ci ha permesso di avere un numero minore di grafici rispetto ad altri approcci e contestualmente poter confrontare semplicemente le prestazioni per singola classe, che è stata reputata come l'informazione migliore per effettuare la valutazione.

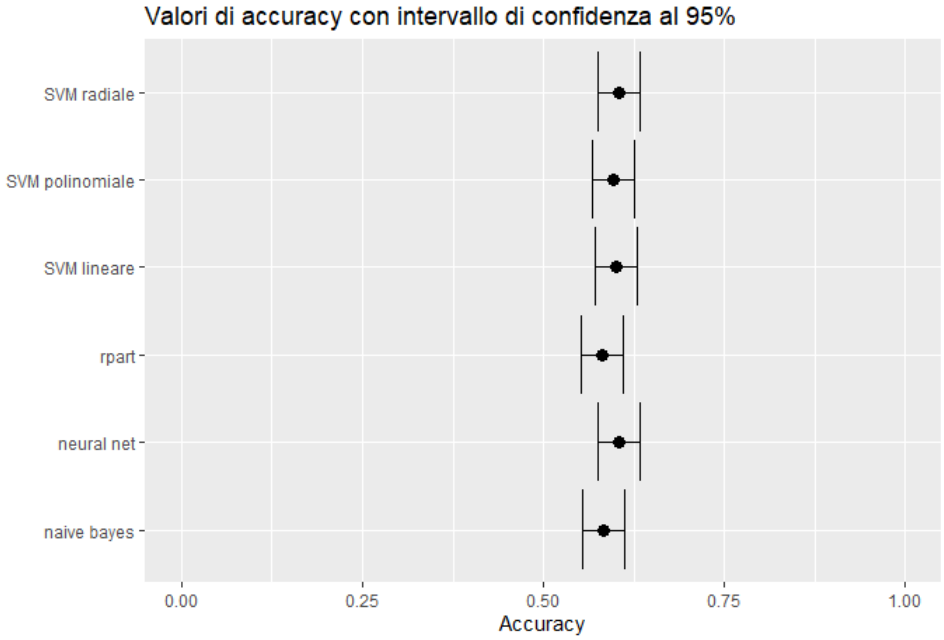

Figure 13: Accuracy dei diversi modelli

Per aggregare le metriche abbiamo optato per l'utilizzo della *macro average* a dispetto della *micro average*, poiché non desideravamo dare più importanza a nessuna classe nonostante lo sbilanciamento delle istanze. Avendo valutato poi singolarmente le prestazione per ogni classe e non solo il valore aggregato, è stato comunque tenuto conto del loro sbilanciamento.

Osservando le metriche di *marco average* si può osservare come le prestazioni medie per tutti i classificatori siano molto simili. I modelli che presentano valori aggregati peggiori sono gli alberi di decisione e, in maniera minore, *SVM* lineare. Al fine di analizzare nel dettaglio le cause che portano a questa differenza con gli altri modelli, è necessario guardare i valori ottenuti per ogni singola classe. In generale, si può affermare che si siano ottenute prestazioni buone sulla classe LOW (le metriche oscillano tra il 70% e 75%), mentre per le altre due classi i valori sono molto meno buoni, con le metriche che oscillano intorno al 50% per entrambe le classi. Si può osservare inoltre come per la classe di minoranza ($HIGH$) alcuni classificatori non siano stati in grado di rilevare alcuna istanza. In particolare, per quanto riguarda l'albero di decisione sono stati ottenuti valori nulli. Questa variazione di valori può essere causata dallo sbilanciamento delle istanze in relazione alle tre classi. Come presentato in precedenza però, anche tecniche come *downsampling* non sono riuscite migliorato la situazione. Si può quindi pensare che queste difficoltà siano anche dovute al fatto che i dati non siano facilmente separabili.

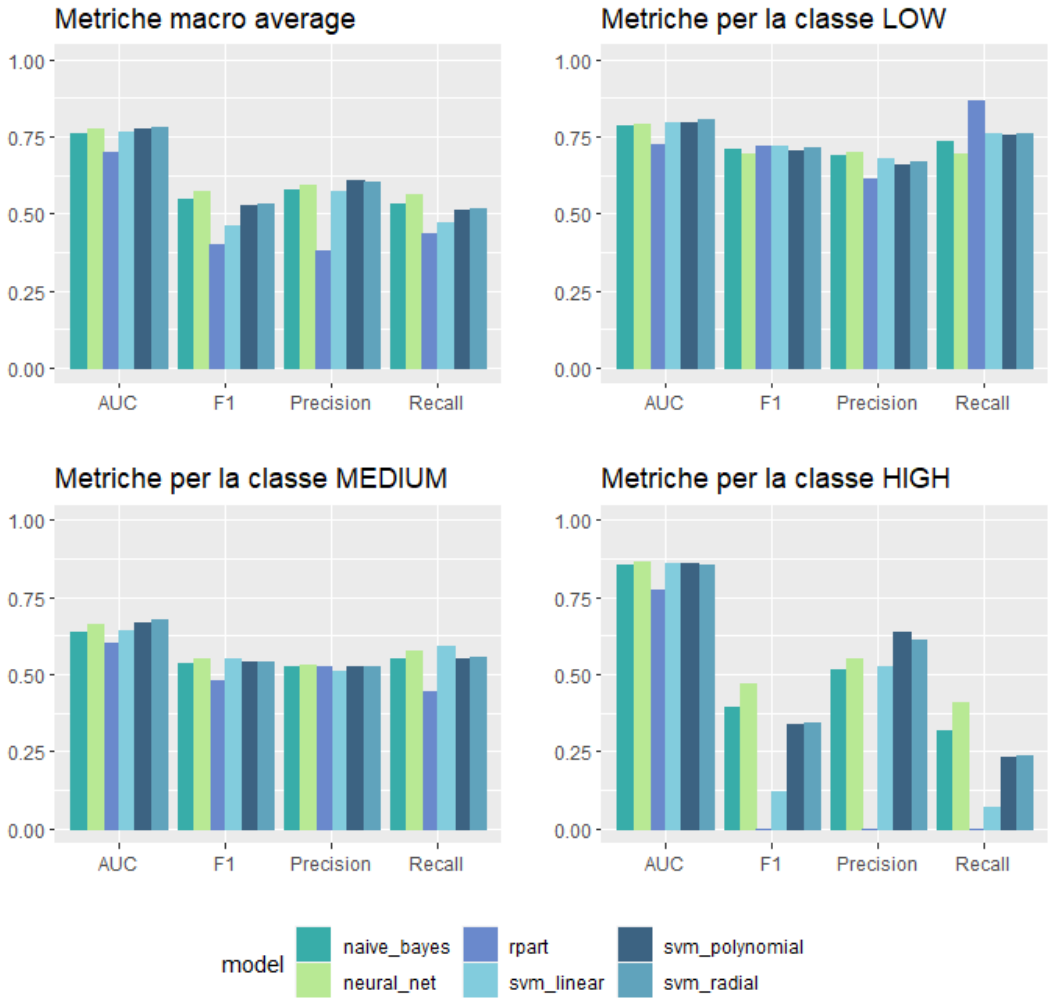

Figure 14: Metriche calcolate raggruppate per classe

Per ogni modello è stato anche generato il grafico relativo alle curve ROC, che rappresenta le 3 curve per le rispettive classi del problema e ulteriori 2 curve relative alle curve calcolate attraverso i valori aggregati di *micro e macro average*. Analizzando i grafici e i valori di *Area Under Curve* presentati nei grafici precedenti (3.3), si può osservare come le curve siano in generale molto simili tra loro e non ci sia grande discrepanza tra i differenti valori di *AUC*. La curva di *rpart* però risulta leggermente peggiore rispetto alle altre, e si può inoltre osservare che gli alberi presentino i valori di *AUC* più bassi tra tutti i modelli.

Per ogni modello inoltre emerge una differenza fra la curva relativa alla classe *MEDIUM* e le altre 4. Questo valore indica la difficoltà dei modelli durante la classificazione di questo tipo di istanze, e anche in precedenza analizzando le metriche calcolate era già stato sottolineato questo aspetto. La classe MEDIUM ha performance peggiori rispetto alla classe HIGH poiché in percentuale tende a sbagliare di più in riferimento al numero di istanze totali.

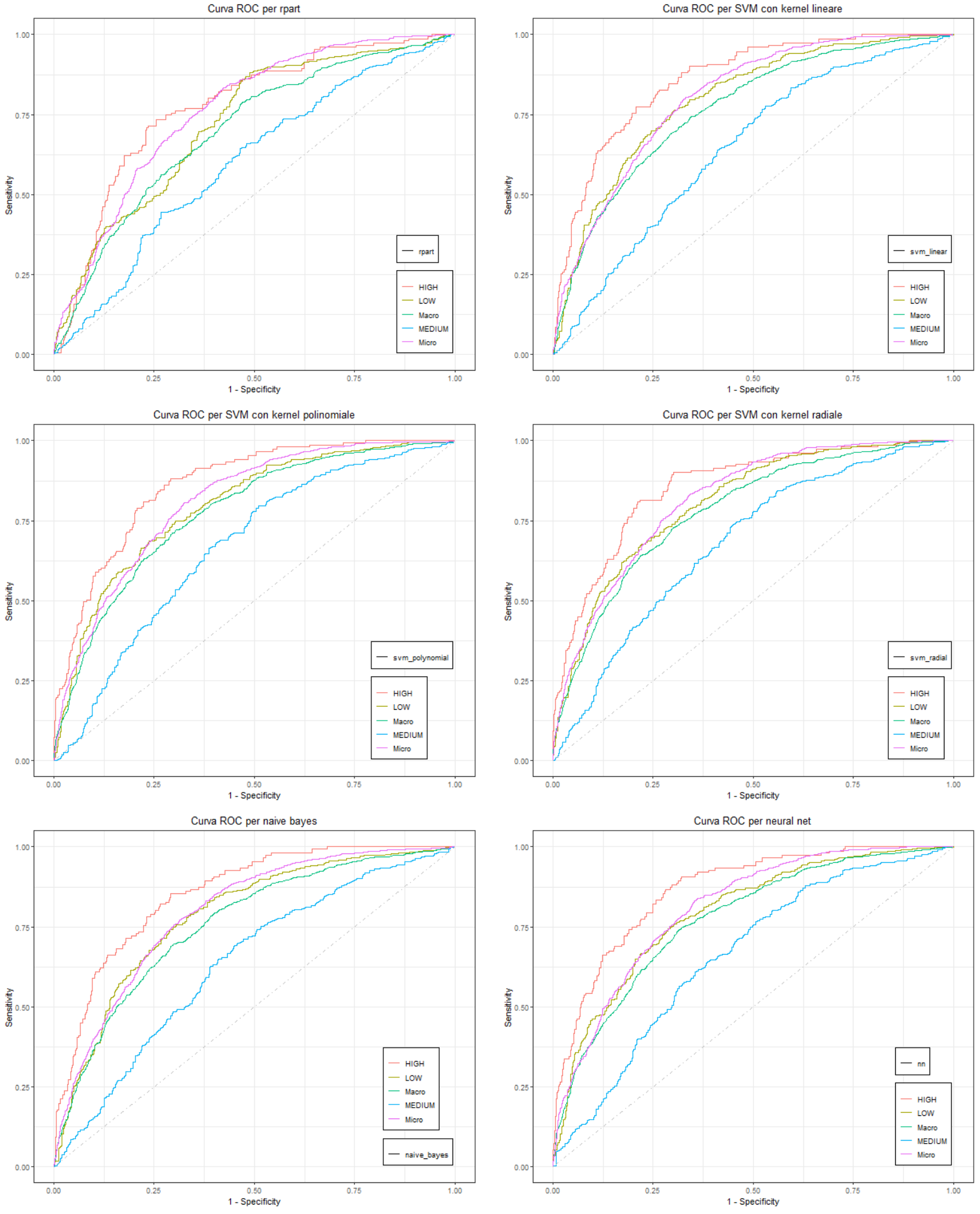

Table 3: Curve ROC per tutti i modelli e per tutte le classi

Un ulteriore analisi delle prestazioni dei modelli si può effettuare considerando i loro tempi di *training* e *prediction*, illustrati nella tabella successiva (4). Come si può osservare dai dati riportati albero decisionale, *naive bayes* e SVM lineare sono i modelli più rapidi (circa 3sec), mentre il più lento in assoluto è SVM polinomiale (27x volte più lento di questi tre). SVM radiale e rete neurale invece presentano tempi più alti rispetto ai più veloci rispettivamente di 4x volte e 8x volte.

Si può notare come ovviamente i modelli più semplici abbiano prestazioni ottime mentre quelli più complessi abbiamo bisogno di maggiore tempo per l'addestramento e la predizione. Si deve comunque considerare che per quanto ci sia una grande discrepanza tra i modelli, avendo utilizzato un dataset con un numero ridotto di istanze e covariate i tempi sono comunque bassi (massimo un minuto e mezzo).

Modello	**Timing**
Albero decisionale	3.09
Naive Bayes	3.05
Svm lineare	3.33
Svm polinomiale	81.91
Svm radiale	13.08
Rete neurale	24.00

Table 4: Tempi di calcolo con diversi metodi di resampling

Considerando tutti gli aspetti presentati è quindi possibile quindi delineare quale siano i modelli che possano essere considerati come i migliori. Oltre alle prestazioni raggiunte, è importante considerare anche il costo per ottenerle, al fine di avere il giusto *trade off* tra tempi e correttezza delle predizioni del modello.

L'albero decisionale è in assoluto il modello più semplice e veloce tra quelli presentati, ma questo si traduce anche in prestazioni non ottimali. Come già presentato in precedenza infatti, non è stato in grado di classificare alcuna istanza di tipo *HIGH*, oltre a non avere prestazioni come gli altri relative a tutte le altre classi. Per questo motivo, si può affermare che, nonostante i suoi pregi, non possa essere considerato in questo caso un buon classificatore.

Naive Bayes è il modello giudicato come migliore tra quelli presentati, poiché si rilevano ottime prestazioni mantenendo però bassa complessità computazionale. Come si può osservare dai grafici infatti, le sue prestazioni sono in linea con quelle degli altri classificatori più complessi ma presenta il tempo in assoluto più basso per addestramento e predizione.

Un modello che ha ottenuto prestazioni inferiori rispetto agli altri è *SVM* lineare, che come gli alberi ha presentato problemi di classificazione per la classe *HIGH*, ma nonostante ciò è un buon *trade-off* tra risultati ottenuti e complessità computazionale. Per questo può essere considerato come un discreto classificatore, nonostanze prestazioni inferiori agli altri modelli.

Per quanto riguarda *SVM* polinomiale invece, presenta ottime prestazioni dal punto di vista delle metriche. Il fattore che penalizza però molto questo classificatore è il tempo di addestramento e predizione, che è in assoluto quello più alto riscontrato tra tutti i modelli analizzati. Possiamo quindi considerare altri classificatori (come ad esempio *naive bayes*,

reti neurali e *SVM* radiale) come migliori poiché, a fronte di prestazioni simili, si ha una complessità computazionale decisamente minore.

SVM radiale e reti neurali possono infine essere considerati entrambi come buoni classificatori. Per quanto riguarda il primo abbiamo ottenuto prestazioni in linea con i modelli migliori, a fronte di una complessità abbastanza bassa. La rete neurale invece, nonostante tempi di addestramento e predizione tra i più alti, è stato il modello che è stato più in grado di effettuare una discriminazione corretta delle istanze *HIGH*. Per i motivi presentati, questi due classificatori possono essere considerati quindi i migliori considerati dopo *naive bayes*.

4 Conclusioni

In questa relazione è stato presentato il lavoro svolto per riguardante l'addestramento di sei diversi modelli di *machine learning*, e la successiva predizione. La fase iniziale ha visto l'esplorazione del dataset, attraverso diversi grafici e l'applicazione della *PCA*. Successivamente sono stati condotti gli esperimenti, attraverso addestramento e predizione utilizzando i vari modelli generati. In seguito, sono stati analizzati i risultati ottenuti al fine di calcolare sia tre diverse metriche (*precision*, *recall*, *f-measure*) che le curve ROC per i vari modelli. Infine sono stati analizzate le metriche e i tempi di esecuzione ed effettuati i confronti tra i vari modelli, per indicare quali secondo noi fossero i modelli migliori. Dal confronto è emerso che il modello migliore è risultato *naive bayes*, sia per i suoi valori ottenuti sia per il suo tempo di esecuzione minimo. Gli altri modelli invece risultati come "più veloci", alberi di decisione e SVM lineare, non hanno invece dato risultati molto buoni (soprattutto il primo). Per quanto riguarda SVM lineare, i suoi valori non sono in linea con quelle ottenute dagli altri modelli ma sicuramente è un ottimo *trade-off* tra prestazioni e tempo. Un altro modello che non si è rivelato come molto adeguato per il problema analizzato è SVM polinomiale, non per i risultati ottenuti ma a causa dell'alta complessità computazionale, che a parità di prestazioni non permette quindi di preferirlo ad altri modelli analizzati. Infine rete neurale e SVM radiale si sono rivelati come un buon *trade-off* tra prestazioni e tempi, e li rendono sicuramente adatti a questo tipo di problema.