Classificazione Multiclasse

Irene Michelotti 319721,Diego Morichelli 305478, Maria Antonietta Longo 308756, Filippo Scaramozzino 312856, Sara Viglietti 315127

***Abstract*—La classificazione, binaria o multiclasse, viene usata in molti ambiti per predire la classe di appartenenza degli oggetti che stiamo studiando. In questo report ci concentreremo sulla creazione di un modello di classificazione multiclasse che suddivida il dataset, costituito da fagioli secchi, in 7 classi distinte corrispondenti alle diverse varietà del fagiolo. In particolare confronteremo quattro tecniche di classificazione (decision tree, random forest, k-nearest neighbors e SVM) cercando di ottenere un modello con accuratezza elevata ma evitando overfitting, poichè l'obiettivo ultimo è ottenere, dato in input un dataset contenente le caratteristiche dei fagioli, una classificazione in gruppi il più precisa possibile per evitare contaminazioni.**

I. Introduzione

I fagioli sono un legume ampiamente prodotto per molteplici scopi alimentari, tuttavia in base alla varietà, identificata da una serie di parametri, cambia la loro qualità e di conseguenza anche il loro valore al mercato. L'obiettivo della nostra analisi è ottenere un metodo efficace e rapido per la classificazione dei fagioli in classi distinte. Il dataset a nostra disposizione è suddiviso in due sottoinsiemi, uno contenente i dati di training che utilizzeremo per addestrare i metodi e l'altro quelli di test che utilizzeremo per testare i metodi sviluppati. Entrambi sono caratterizzati da 16 attributi diversi e dall'attributo classe che, nel caso dei dati di training, utilizzeremo per ottenere le informazioni che caratterizzano le singole varietà, mentre nel caso dei dati di test verrà usato per determinare l'accuratezza delle previsioni dei metodi usati.

II. Analisi dei dati

Nel corso della nostra analisi metteremo a confronto vari metodi di classificazione al fine di capire quale sia il migliore per il dataset di cui disponiamo.
Prima di creare un modello, analizziamo i dati quali a nostra disposizione, eseguiamo quindi exploratory data analysis e attuiamo procedure di *pre-processing*. Disponiamo di due dataset: quello di *training* che useremo per addestrare il modello e quello di *test* che utilizzeremo per capire quanto è accurato il singolo metodo di classificazione e quanto è performante rispetto ad altri.
In particolare il *Dry Bean Dataset Trainig* è composto da 2926 osservazioni, mentre il *Dry Bean Dataset Test* ne contiene 3654. Notiamo che entrambi i dataset sono bilanciati, ovvero il numero di istanze di fagioli per ciascuna classe è simile.
Per entrambi i dataset, prima di ogni procedura di analisi e pulizia, abbiamo le seguenti features:

1) Area (A): Area del fagiolo.
2) Perimeter (P): Circonferenza del fagiolo.
3) Major axis length (L): Lunghezza dell'asse maggiore.
4) Minor axis length (l): Lunghezza dell'asse minore.
5) Aspect ratio (K): Definisce la relazione tra L ed l.
6) Eccentricity (Ec): Eccentricità dell'ellisse che forma il fagiolo.
7) Convex area (C): Numero di pixel nel più piccolo poligono convesso che contiene l'area del fagiolo.
8) Equivalent diameter (Ed): Diametro di un cerchio avente la stessa area del fagiolo.
9) Extent (Ex): Rapporto tra i pixel nella bounding box rispetto all'area del fagiolo.
10) Solidity (S): Nota anche come convessità. Rapporto tra i pixels nel guscio convesso rispetto a quelli che si trovano nei fagioli.
11) Roundness (R): Calcolato con la formula $\frac{4\pi A}{P^2}$
12) Compactness (CO): Misura della rotondità $\frac{Ed}{L}$
13) ShapeFactor1 (SF1)
14) ShapeFactor2 (SF2)
15) ShapeFactor3 (SF3)
16) ShapeFactor4 (SF4)
17) Classe

Per aiutarci a comprendere meglio i dati abbiamo utilizzato delle rappresentazioni grafiche, prima raggruppando i dati per classe, e dopo analizzando le distribuzioni numeriche degli attributi. Riportiamo due esempi.

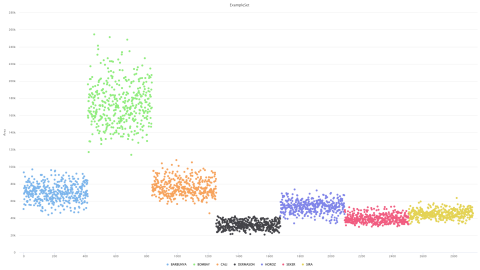

Fig. 1: Distribuzione Area delle classi

Fig. 2: Distribuzione Compactness delle classi

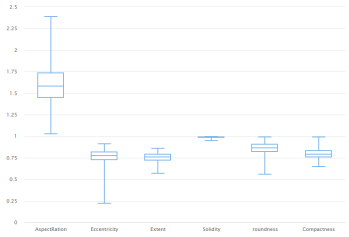

Fig. 3: BoxPlot dei dati

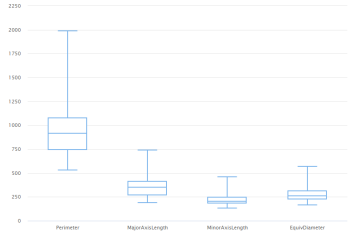

Fig. 4: BoxPlot dei dati

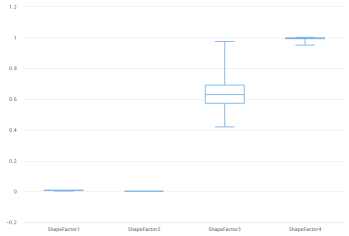

Fig. 5: BoxPlot dei dati

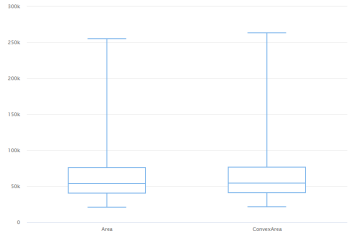

Fig. 6: BoxPlot dei dati

Analizzando i BoxPlot notiamo che le diverse features hanno ordini di grandezza molto differenti tra loro, questo ci fa capire che condurre uno scaling dei dati sarà fondamentale prima di procedere con l'addestramento dei metodi.

Distribuzione Classi		
Nome Classe	Count	Percentuale
SIRA	418	14.29%
CALI	418	14.29%
SEKER	418	14.29%
BOMBAY	418	14.29%
BARBUNYA	418	14.29%
DERMASON	418	14.29%
HOROZ	418	14.29%

Distribuzione dei Dati				
Attributo	Min	Max	Media	SD
Area	20786	254616	69956.8	46063.0
AspectRatio	1.0	2.3	1.5	0.2
Compactness	0.6	0.9	0.8	0.06
ConvexArea	21057	263261	70937.0	46724.1
Eccentricity	0.2	0.9	0.7	0.09
EquivDiameter	162.6	569.3	286.0	85.1
Extent	0.5	0.8	0.7	0.05
MajorA.Length	189.8	738.8	363.4	115.2
MinorA.Length	132.1	460.2	227.8	66.6
Perimeter	530.8	1985.4	971.8	296.8
ShapeFactor1	0.003	0.01	0.006	0.001
ShapeFactor2	0.001	0.004	0.002	0.001
ShapeFactor3	0.4	0.9	0.6	0.1
ShapeFactor4	0.9	1.000	0.9	0.005
Solidity	0.9	0.9	0.9	0.005
Roundness	0.5	0.9	0.8	0.06

I primi 16 attributi rappresentano i predittori e sono a valori numerici, mentre la feature Classe indica le 7 possibili classi di appartenenza per ognuno dei fagioli: Seker, Barbunya, Bombay, Cali, Dermason, Horoz e Sira.
Ognuna di queste si differenzia dalle altre per i valori assunti dalle features appena descritte e l'obiettivo è quello di capire quale sia la classe di appartenenza di una nuova osservazione. Dobbiamo quindi fare predizione sulla classe di appartenenza per una nuova osservazione.
L'analisi è stata condotta in parte attraverso *Rapid Miner Studio* e in parte attraverso *python* laddove quest'ultimo semplificasse la leggibilità dei risultati ad esempio per SVM con tecnica 1 vs all.
In primo luogo abbiamo rimosso i duplicati (uno era presente nel *Training set*), controllato che non ci fossero valori mancanti e osservazioni con valori negativi corrispondenti alle varie *features* numeriche: infatti, per come sono definite le *features* di questo problema, nessuna di loro può essere negativa.
Inoltre, dal momento che stiamo lavorando con dati numerici e i metodi che addestreremo spesso utilizzano metriche, è importante fare uno scaling dei dati. Abbiamo normalizzato i valori degli attributi numerici, utilizzando una standardizzazione, per ottenere dei dati con distribuzione a media 0 e varianza 1.
La ragione per cui abbiamo pensato di riportare come vengono definite le features a livello matematico è per porre l'attenzione sul fatto che molte di queste sono derivate dalle altre, ad esempio la *Compactness* (12), che si calcola come $\frac{Ed}{L}$ non aggiunge nulla di nuovo alle features precedenti: qualora ne avessimo bisogno per qualche motivo, potremmo calcolarla direttamente a partire da perimetro ed area che sono già presenti tra le features. Alla luce di ciò abbiamo pensato che fosse inutile condurre l'analisi mantenendo tutte le features ed è per questo che abbiamo implementato e confrontato tra loro due procedure possibili per ridurre la dimensionalità: analisi della correlazione e Principal Component Analysis (PCA).
Attraverso *Rapid Miner Studio* abbiamo eliminato gli attributi con correlazione maggiore del 95% (in valore assoluto) ciò

ha fatto sì che 7 delle 16 variabili fossero eliminate. La PCA ha fornito un risultato analogo: con sole 5 componenti viene catturata il 98% della varianza. Il modello di PCA applicato al training set è stato applicato al test set in fase di classificazione dei dati.
Da questo momento in poi useremo sempre i dati con il numero ridotto di features.

III. Metodi

Per analizzare il dataset abbiamo utilizzato 4 metodi diversi di classificazione:

- Decision tree
- Random Forest
- KNN
- SVM

Per il tuning degli iperparametri abbiamo utilizzato la k-fold cross validation con k=10, utilizzando soltanto il training set. Per fare tuning degli iperparametri è stato utilizzato Python.

A. DECISION TREE

Il decision tree è il classificatore più semplice presente in letteratura, è di facile interpretazione e visualizzazione, tuttavia non è incrementabile ed è sensibile ai dati rumorosi.
Gli algoritmi di splitting implementati in questo modello si basano su una strategia greedy, che consiste nella ricerca ad ogni passo della migliore feature per splittare i dati, in funzione del criterio che si sceglie di seguire.
Per quest'analisi abbiamo optato per una scissione binaria.
Gli altri parametri da settare per un albero decisionale sono legati al criterio di arresto e di splitting dei nodi.
Oltre ad essere un classificatore interpretabile fornisce anche la feature importance, permettendo così di capire quali siano le caratteristiche del dataset veramente utili.
Abbiamo fatto leva su questa proprietà in modo tale da aiutare le aziende a risparmiare tempo e risorse per la misurazione dei fagioli.

B. RANDOM FOREST

Come secondo metodo di classificazione è stato usato il Random Forest, che consiste nell'addestramento di più alberi di decisione, ognuno con un subset di cardinalità $|m|$ di feature casuali, e restituisce la classe predetta utilizzando il majority voting.

Gli iperparametri da settare in questo caso sono il numero di feature da trattare per ogni singolo albero e il numero di alberi decisionali.
Questo modello è più robusto e accurato dell'albero decisionale, poichè essendoci più alberi nel caso di predizioni errate gli altri bilanciano il risultato; tuttavia, questo porta anche a delle predizioni non interpretabili dal momento che si basa su un numero elevato di alberi.

C. KNN

Un approccio completamente diverso rispetto ai due precedenti si ottiene usando il k-nearest neighbors.
Per ogni nuovo record da classificare viene calcolata la similarità (interpretabile come distanza) con tutti i dati di training, a questo punto si identificano i "K" elementi più vicini e si assegna al record l'etichetta di classe basandosi sulla classe prevalente. Il k-nearest neighbor gode di incrementabilità.

Gli iperparametri su cui dobbiamo fare tuning sono:

- K : con un valore troppo elevato di k la classificazione può includere punti di altre classi, se invece settassimo k ad un valore troppo piccolo otterremo un dataset sensibile ai dati rumorosi e agli outliers.
- Per misurare la similarità abbiamo utilizzato la distanza di grado g :

$$d(p,q) = (\sum_{i=1}^{n} |p_i - q_i|^g)^{\frac{1}{g}}$$

  si richiede quindi un tuning di g.
- Il peso da dare ai dati con distanza differente, più alta per quelli più vicini e minore per quelli più distanti.

D. SVM

Questo algoritmo, tramite una ottimizzazione regolarizzata, cerca un iperpiano che separi le diverse classi.
L'SVM è un modello black box, dunque non è interpretabile, inoltre non è incrementabile, ma ha buone performance a livello di accuratezza e efficienza.
Nel nostro caso avendo 7 classi ci siamo affidati al SVC in Python che performa per ogni classe la tecnica "1 vs all" e ottiene 7 iperpiani che separano le differenti classi.
Abbiamo provato 2 tipi di SVM, lineare e polinomiale.

Gli iperparametri su cui fare tuning sono:

- C: parametro di regolarizzazione
- p: grado del polinomio nel caso polinomiale

IV. Risultati sperimentali

A. DECISION TREE

Dalla validazione otteniamo che:

- la profondità massima(max-depth) dell'albero è 15
- il miglior criterio per lo splitting è l'information gain, ovvero il guadagno di informazione quando la misura di impurità usata è l'entropia

Abbiamo ottenuto quindi la feature importance, come descritto nella sezione metodi, e possiamo visualizzare i quattro attributi più importanti per effettuare lo splitting, e la loro relativa importanza nel grafico seguente:

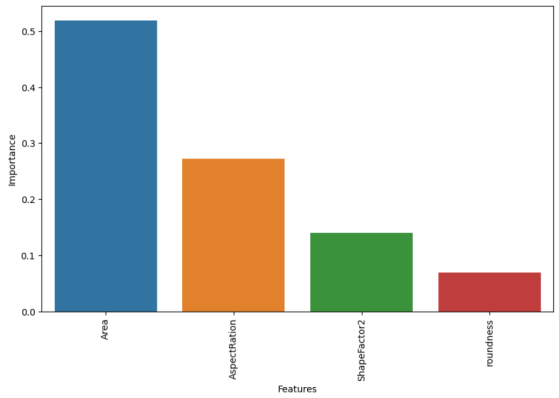

Fig. 7: Feature importance

B. RANDOM FOREST

Nel random forest abbiamo seguito lo stesso iter del decision tree, cercando di trovare le features fondamentali.

Dalla validazione otteniamo:

- Oltre i 100 alberi non vi è alcun miglioramento delle performance e già attorno ai 50 si inizia a osservare un plateau
- Come per il decision tree, risulta dalla validazione che questo algoritmo è estremamente sensibile alla profondità ed è per questo che abbiamo settato la massima profondità a 15
- Il numero ottimale di feature da oscurare per ogni albero è proprio 1, quindi questo dataset viene analizzato molto bene da alberi che trattano una caratteristica alla volta.

C. KNN

Dalla validazione si ottiene che:

- Il miglior metodo di pesare le istanze è quello di dare un peso inversamente proporzionale alla distanza
- Gli ordini p di distanza sono i più semplici (p=1, p=2)
- Il numero K ottimale è 8 o 9.

D. SVC

Nel polinomiale cerchiamo, validando, il miglior parametro regolarizzatore e grado polinomiale p ottenendo i seguenti risultati:

- Il miglior grado è sempre 1, prova del fatto che un svc lineare fitta meglio i dati
- Il parametro regolarizzatore è sempre C=1

E. RISULTATI FINALI

Analizziamo i risultati ottenuti con i diversi metodi. Per la nostra analisi utilizzeremo le confusion matrix in cui possiamo vedere la performance dei metodi e l'accuracy, una metrica consigliata quando il dataset è bilanciato come nel nostro caso, che indica la frazione di dati classificati nel modo corretto.

Risultati Confusion Matrix		
Metodo	Accuracy	Classification Error
Decision Tree	97.40%	2.60%
Random Forest	98.16%	1.84%
KNN	98.11%	1.89%
SVM	94.19%	5.81%

Infine, possiamo affermare che la tipologia di fagioli che più viene predetta nelle classi sbagliate è la Sira, mentre la tipologia Bombay è predetta correttamente al 100%. Questi risultati sono analoghi per tutti i metodi.

V. CONCLUSIONI

Come possiamo osservare dai risultati delle confusion matrix, tutti i metodi applicati hanno avuto ottime performance. Cerchiamo di valutare la nostra analisi in un setting di business. Se desideriamo creare un modello interpretabile da fornire ad un possibile cliente, il migliore è il decision tree. Il random forest offre una robustezza maggiore, però è meno interpretabile e quindi potrebbe essere uno svantaggio per un possibile cliente non esperto. Il metodo SVM risulta essere il meno performante anche se presenta risultati soddisfacenti, la sua non interpretabilità però lo rende meno fruibile per i clienti. Il metodo KNN presenta ottimi risultati, e una buona interpretabilità. La nostra analisi, a fini aziendali, potrebbe migliorare significativamente il processo tramite cui si possono identificare in modo veloce e automatizzato i diversi tipi di faglioli. Identificare la classe corretta dei fagioli può essere utile per scopi logistici, in quanto la conservazione nei magazzini e la loro spedizione verso clienti potrebbe avvenire in modo diverso per ogni tipo di fagiolo, e per scopi di previsione di rendimento dato dalla vendita dei fagioli, in quanto ogni tipo di fagiolo potrebbe essere venduto a prezzo diverso.

VI. CONTRIBUTI

Per quanto riguarda i contributi personali, Irene Michelotti (319721) si è occupata dell'introduzione del problema esaminando: il paper originale, le features che componevano il dataset e lo scopo finale della nostra analisi. Maria Antonietta Longo (308756) si è occupata del pre-processing, ovvero della preparazione del dataset che è stato successivamente utilizzato per addestrare i metodi, analizzando la correlazione tra features e applicando PCA. Filippo Scaramozzino (312856) si è occupato della exploratory data analysis, successivamente dell'identificazione dei metodi di classificazione adatti al nostro problema, e in fine, delle possibili applicazioni di business. Diego Morichelli (305478) si è occupato della parte di validazione dei metodi, al fine di trovare i parametri corretti con cui addestrare i metodi, e della parte in Python in cui abbiamo sviluppato il metodo SVM. Sara Viglietti (315127) si è occupata dell'analisi dei risultati dei metodi tramite le confusion matrix, e di analizzare quali fossero i metodi migliori e classi predette meglio rispetto alle altre. L'analisi svolta e la stesura di questo lavoro è stata sempre condotta con la piena partecipazione e collaborazione di tutti i membri del gruppo al fine di garantire una divisione dei compiti uniforme e inclusiva per ogni componente.