Analisi e classificazione di tipologie di semi di fagioli

Marco De Ramundo – Università degli studi di Brescia

ARTICOLO DI RIFERIMENTO *"Multiclass classification of dry beans using computer vision and machine learning techniques"*

LINK COLAB NOTEBOOK : colab.research.google.com/drive/1R80-_IFWhBZfiAWtDj-XkJ4-2OTvuyun?usp=sharing

ABSTRACT

Obiettivo del progetto è quello di analizzare il dataset proposto, verificare i risultati ottenuti da parte dell'autore dell'articolo e infine proporre nuovi modelli che possono migliorare le performance di classificazione. L'articolo di riferimento è "*Multiclass classification of dry beans using computer vision and machine learning techniques*" di Murat Koklu e Ilker Ali Ozkan, ricercatori del Department of Computer Engineering, Selcuk University, Turchia. L'oggetto di analisi riguarda la classificazione tramite caratteristiche fisiche e geometriche della forma dei fagioli in modo da riconoscerne la rispettiva tipologia.

DATASET

Il progetto in questione si basa sul dataset proposto dagli autori dell'articolo di riferimento e disponibile pubblicamente sul sito dell' UCI. Il dataset è una raccolta di caratteristiche fisiche di semi di fagioli i quali sono stati classificati secondo la loro tipologia. Il lavoro di raccolta dei dati è stato manuale per quanto riguarda la classificazione dei vari semi mentre per le proprietà fisiche si è fatto ricorso all'acquisizione e analisi delle immagini. Fotografando una certa quantità di semi sono riusciti a ricavare tramite l'elaborazione delle immagini alcune misure che hanno poi popolato il dataset analizzato.

Il numero di classi in questione sono 7, sono le tipologie di fagioli più diffusi in Turchia (paese di origine dell'articolo). Nello specifico le tipologie sono:

- ***Cali***: semi di colore bianco, leggermente tondi e poco più grandi rispetto a quelli secchi con una forma classica a fagiolo.
- ***Horoz***: sono semi lunghi, di forma cilindrica, di colore bianco e in genere di dimensione media.
- ***Dermason***: sono semi più pieni, di colore bianco e tondi alle estremità.
- ***Seker***: semi grandi, colore bianco e di forma rotonda.
- ***Bombay***: semi di colore bianco, molto grandi e di forma ovale allungata.
- ***Barbunya***: semi di colore beige con delle piccolo macchie o striscie rosse, sono dei semi grandi e di forma ovale, quasi tondeggiante.
- ***Sira***: sono semi piccoli, di colore bianco, con una faccia piatta e l'altra più tondeggiante.

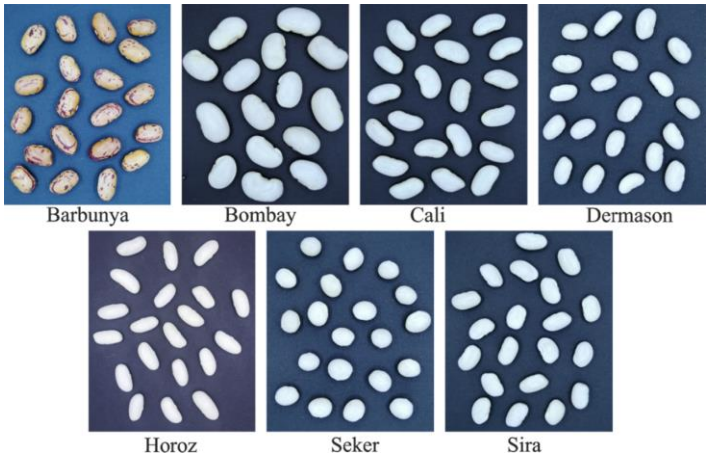

Figura 1 - tipologie dei semi di fagioli trattati

Per ogni seme misurato sono stati calcolati 16 caratteristiche fisiche che descrivono in qualche modo la forma del fagiolo. I valori misurati dipendono dal numero di pixel che disegnano il seme nell'immagine e le features si basano su questa pseudo unità di misura. Molte delle features sono ricavate da una serie di calcoli che prevedono le features base. Le feature considerate per popolare il dataset sono:

- ***Area*** (A): Area del seme nell'immagine. Misurando contando il numero di pixel che compone il seme nell'immagine. Nel dataset segnato come valore intero.
- ***Perimeter*** - perimetro (P): Lunghezza del bordo del seme, misurando contando i pixel di bordo nell'immagine.
- ***Major axis length*** – lunghezza asse maggiore (L): La distanza massima tra due punti estremi di un seme.
- ***Minor axis length*** – lunghezza asse minore (l): La distanza massima tra due punti estremi che risultano perpendicolari all'asse maggiore.
- ***Aspect ratio*** (K): Definisce la relazione tra i due assi L e l.
  - K = L / l
- ***Eccentricity*** - eccentricità (Ec): Eccentricità dell'ellisse costruito avente come assi L e l.
- ***Convex area*** – area convessa (C): numero di pixel all'interno del più piccolo poligono convesso che contiene l'immagine del seme. Nel dataset segnato come valore intero.
- ***Equivalent diameter*** – diametro equivalente (Ed): Diametro relativo al cerchio avente la stessa area del seme.
  - Ed = $\sqrt{(4 * A)/\pi}$
- ***Extent*** - estensione (Ex): rapporto tra il numero di pixel dell'area A rispetto all'area del rettangolo che racchiude il seme.
  - Ex = A / $A_{rettangolo}$
- ***Solidity*** - solidità (S): oppure convessità, rapporto tra il numero di pixel dell'area A rispetto all'area convessa C.
  - S = A / C
- ***Roundness*** – rotondità (R):
  - R = 4πA / P^2
- ***Compactness*** – compattezza (CO):
  - CO = Ed / L
- ***ShapeFactor1*** – fattore di forma 1 (SF1):
  - SF1 = L / A
- ***ShapeFactor2*** – fattore di forma 2 (SF2):
  - SF2 = l / A
- ***ShapeFactor3*** – fattore di forma 3 (SF3):

- SF3 = 4A / L^2* π
- ***ShapeFactor4*** – fattore di forma 4 (SF4):
  - SF4 = 4A / L*l*π

Come si può notare molte features sono derivate da alcune features base tramite una serie di equazioni. Questa caratteristica risulterà importante nel fase di feature engineering in quanto come vedremo il dataset presenta numerose collinearità.

Ogni istanza registrata all'interno del dataset è segnato in modo completo e non ci sono valori mancanti. Il dataset da quel punto di vista è completo e non è necessario alcun tipo di soluzione per stimare o completare le informazioni mancanti. Il numero di record per classe non è omogeneo, ci sono classi con un numero di record molto maggiore rispetto ad altre. Nella tabella sono riportati il numero di istanze per classe.

Classe	Numero istanze
Cali	1630
Horoz	1928
Dermason	3546
Seker	2027
Bombay	522
Barbunya	1322
Sira	2636
Totale	13611

Tabella 1 - numero istanze per classe contenute nel dataset

Il fatto di avere un numero non omogeneo di campioni per classe deriva dal fatto che i ricercatori che hanno pubblicato l'articolo hanno considerato una quantità di semi avente peso costante per tutte le classi. In alcuni casi durante l'addestramento dei modelli questo fattore è stato preso in considerazione per evitare che il modello fosse influenzato maggiormente da una specifica rispetto a un'altra.

FEATURE ENGINEERING

Prima di procedere con l'addestramento di una serie di modelli di classificazione è stata affrontata una fase di feature engineering con lo scopo di analizzare le varie features, valutare eventuali correlazioni e verificare se ci sia la possibilità di scartare alcune features oppure modificare il dataset per poter rendere più semplice il modello o l'addestramento senza perdere troppo in termini di efficacia. Come citato prima durante la descrizione del dataset, praticamente tutte le features fanno riferimento a caratteristiche fisiche dei semi, nello specifico caratteristiche geometriche che tendono a descrivere la forma dei fagioli. Infatti caratteristiche nominali come poteva essere il colore del seme nel lavoro dei ricercatori non sono state prese in considerazione. L'idea dietro la creazione del dataset è quella di raccogliere tutta una serie di caratteristiche che mostrano la differenza di una tipologia rispetto a un'altra. Il problema di questo metodo è che molte delle features sono quindi derivate da alcune features base e perciò i valori saranno dipendenti tra loro.

Per questo motivo è stata introdotta una variabile `COLLINEAR_THRESHOLD` che verrà utilizzata per il calcolo delle collinearità tra le features. Se due features hanno una correlazione troppo alta questa viene evidenziata e sarà possibile escludere una delle due feature per alleggerire il dataset mantenendo prestazioni comunque positive. Il valore della `COLLINEAR_THRESHOLD` è stato impostato a 0.99, valore di per sé molto alto ma dovuto al fatto che durante l'analisi molte delle features sono correlate tra loro. Si è preferito evitare di rimuovere troppe feature con il rischio di perdere in prestazioni ma piuttosto tenere una soglia elevata per

poter rimuovere features che sicuramente non avrebbero portato un valore aggiunto al modello. Inizialmente tale valore è stato impostato a 0.90, poi alzato a 0.95 e infine a 0.99. Sotto nella tabella si possono vedere le features che sarebbero state considerate collineari a seconda della soglia impostata.

0.90	**0.95**	**0.99**
Area	Area	Area
AspectRatio	Compactness	Compactness
Compactness	ConvexArea	ConvexArea
ConvexArea	EquivDiameter	EquivDiameter
Eccentricity	***MajorAxisLength***	Perimeter
EquivDiameter	***MinorAxisLength***	ShapeFactor3
MajorAxisLength	Perimeter	
MinorAxisLength	ShapeFactor3	
Perimeter		
ShapeFactor3		

Tabella 2 - features collineari al variare della soglia

Come si può vedere al diminuire della soglia di collinearità aumentano le features collineari. Nel caso della soglia 0.9 si hanno 10 features su 16 correlate tra loro (62,5%); nel caso della soglia impostata a 0.95 8 features su 16 correlate (50%); infine per la soglia impostata a 0.99 si hanno 6 features su 16 collineari (37,5%). Il motivo per cui si è deciso come valore di soglia 0.99 è dato dal fatto che si è preferito non escludere features che derivano da misurazioni dirette e non da calcoli tramite qualche formula. Infatti qualora si fosse impostato come soglia il valore 0.95 le features degli assi, calcolate tramite misurazione dell'immagine, non sarebbero state considerate. A maggior ragione nel caso di 0.9 si sarebbero aggiunte l'eccentricità e l'aspect ratio.

Ovviamente le features non possono essere scartate tutte in blocco, è necessario che almeno una rimanga per poter garantire l'informazione che sarà utile in fase di addestramento. Nel nostro caso si è deciso di mantenere come feature l'Area, valore base che caratterizza direttamente la descrizione del seme e che per altro è uno dei parametri alla base di altre caratteristiche. La scelta dell'area non è obbligata anzi sarebbe possibile procedere all'addestramento con una feature differente e data la collinearità i risultati non sarebbero molto distanti da quelli ottenuti.

Pertanto le features che sono state ignorate con l'addestramento tramite il dataset semplificato sono:

- *Compactness*
- *ConvexArea*
- *Perimeter*
- *EquivDiameter*
- *ShapeFactor3*

Per completezza comunque i modelli sono stati testati sia con il dataset originale che con quello semplificato e saranno commentati i risultati ottenuti.

Tali correlazioni tra le varie features sono state messe in risalto anche tramite l'utilizzo del Data Profiling, strumento fornito dalla libreria panda. Dalla analisi del dataset si evince come all'intero dataset ci siano in generale forti correlazioni tra le tutte features, che pertanto indicano come le varie variabili sono tutte fortemente legate fra loro. Questa è forse una delle debolezze del dataset dovuto a come è stato pensato e costruito. In ogni caso come mostreremo successivamente e come dimostrato nell'articolo è stato comunque possibile ottenere buoni risultati per i modelli classificatore.

SUPPORT VECTOR MACHINE

Il primo metodo adottato per ricavare il modello classificatore è il support vector machine, metodo in grado di separare le classi nello spazio delle ipotesi tramite dei vincoli separatori quali per esempio degli iperpiani. Questo metodo è applicabile dato dal fatto che tutte le features sono rappresentate come valori numerici. Tale metodo è stato scelto anche dai ricercatori come uno dei possibili modelli classificatori. Nell'articolo si è fatto riferimento alle varie tipologie di SVM disponibili, dalle lineari alle non lineari. Gli autori hanno proposto un modello non lineare basato su una funzione kernel polinomiale. In particolare hanno quindi deciso di costruire un classificatore SVM cubico e le prestazioni ottenute sono mostrate nella seguente matrice di confusione.

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	1919	9	0	1	1	48	49
BARBUNYA	11	1221	1	60	6	23	0
BOMBAY	0	0	522	0	0	0	0
CALI	4	42	0	1549	24	11	0
HOROZ	0	6	0	27	1830	47	18
SIRA	22	9	0	5	33	2289	278
DERMASON	38	1	0	0	4	157	3346

Tabella 3 - matrice di confusione modello support vector machine proposto nell'articolo

Le prestazioni di tale classificatore sono particolarmente buone e inoltre è il miglior modello che sono riusciti a costruire. Infatti il modello ha un'accuratezza del 93,13% ed error rate al 6,87% mentre la precisione di assesta al 94,45% e F1-score si aggira al 94,23%.

Da questi risultati son partito per sviluppare un mia versione del classificatore tramite SVM. Perciò ho eseguito dei training con dataset completo e semplificato sia con un modello SVM non lineare sia con l'intenzione di verificare anche con funzione kernel lineare. Inizialmente non ho apportato modifiche particolari al dataset in modo da vedere come il metodo si apprestava ai dati originali.

Come primo tentativo ho utilizzato il support vector classifier con la funzione kernel di default, ovvero la rbf, la radial basis function. I risultati ottenuti come prevedibile non sono particolarmente entusiasmanti. Infatti sia con il dataset semplificato che con quello completo l'accuratezza non si migliora rispetto al 63%.

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	530	0	0	0	46	752	699
BARBUNYA	0	63	0	894	321	44	0
BOMBAY	0	0	520	2	0	0	0
CALI	0	43	0	1440	142	5	0
HOROZ	54	36	0	102	1142	561	33
SIRA	319	0	0	0	232	1946	139
DERMASON	410	0	0	0	0	132	3004

Tabella 4 - matrice di confusione primo modello support vector machine con kernel rbf su dataset completo

Accuratezza: 63,51% - Precisione: 63% - F1: 59%

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	533	0	0	0	48	751	695
BARBUNYA	0	67	0	888	321	46	0
BOMBAY	0	1	520	1	0	0	0
CALI	0	53	0	1435	136	6	0
HOROZ	54	44	0	99	1130	565	36
SIRA	323	0	0	0	234	1938	141
DERMASON	407	0	0	0	0	135	3004

Tabella 5 - matrice di confusione primo modello support vector machine con kernel rbf su dataset semplificato

Accuratezza: 63,38% - Precisione: 61% - F1: 59%

Quello che si può notare da questa prova iniziale è come il dataset ha bisogno di una fase di preprocessing per il dataset prima che venga utilizzato per l'apprendimento. Molte delle istanze vengono classificate in modo errato riducendo quindi le prestazioni del modello così costruito. D'altro canto si può vedere come una delle classi si distingue dalle altre, la classe BOMBAY, in quanto è facilmente distinguibile dalle altre. Questo potrebbe non stupirci in quanto i semi di questa classe sono in genere più grandi rispetto agli altri e pertanto distinguibili in maniera abbastanza semplice. Ciononostante questa rimane l'eccezione: molte delle classi si differenziano per la forma e non per la grandezza e in questo caso il classificatore per come è costruito non è in grado di poter garantire le prestazioni sperate.

Visto questi risultati preliminari ho evitato di controllare il classificatore svm lineare per poter prima effettuare un'operazione di scaling del dataset. Come vedremo anche in altri metodi, la normalizzazione del dataset porterà numerosi benefici alle prestazioni del modello.

Il dataset normalizzato viene salvato nella variabile dataset_scaled. Anche in questo caso la collinearità con soglia al 99% risulta medesima con il dataset originale. Quindi il dataset semplificato dopo la normalizzazione avrà le stesse colonne del dataset semplificato descritto precedentemente. Perciò è stato possibile ripetere i training con i medesimi modelli, ovvero classificatore support vector machine con kernel rbf e con kernel lineare. Di seguito sono mostrati i risultati assieme alle matrici di confusione

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	1935	9	0	0	0	57	26
BARBUNYA	10	1212	0	68	5	27	0
BOMBAY	0	0	522	0	0	0	0
CALI	3	35	0	1558	22	12	0
HOROZ	0	3	0	24	1849	36	16
SIRA	28	7	0	3	36	2338	224
DERMASON	56	0	0	0	3	220	3267

Tabella 6 - matrice di confusione modello support vector machine con kernel rbf su dataset normalizzato

Accuratezza: 93,16% - Precisione: 94% - F1: 94%

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	1901	17	0	1	1	79	28
BARBUNYA	11	1177	0	82	5	47	0
BOMBAY	0	0	522	0	0	0	0
CALI	2	36	0	1540	29	23	0
HOROZ	0	3	0	29	1825	53	18
SIRA	31	5	0	6	35	2350	208
DERMASON	61	1	0	0	8	292	3184

Tabella 7- matrice di confusione modello support vector machine con kernel lineare su dataset normalizzato

Accuratezza: 91,83% - Precisione: 93% - F1: 93%

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	1937	10	0	0	0	53	27
BARBUNYA	9	1218	0	64	4	27	0
BOMBAY	0	0	522	0	0	0	0
CALI	3	35	0	1557	23	12	0
HOROZ	0	2	0	23	1853	34	16
SIRA	25	7	0	2	36	2340	226
DERMASON	55	0	0	0	3	214	3274

Tabella 8 - matrice di confusione modello support vector machine con kernel rbf su dataset normalizzato e semplificato

Accuratezza: 93,31% - Precisione: 94% - F1: 94%

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	1899	17	0	2	1	79	29
BARBUNYA	12	1177	0	83	5	45	0
BOMBAY	0	0	522	0	0	0	0
CALI	2	37	0	1538	29	24	0
HOROZ	0	3	0	28	1827	52	18
SIRA	33	6	0	7	36	2337	217
DERMASON	62	1	0	0	8	308	3167

Tabella 9 - matrice di confusione modello support vector machine con kernel lineare su dataset normalizzato e semplificato

Accuratezza: 91,60% - Precisione: 93% - F1: 93%

I modelli svm con kernel rbf raggiungono un'accuratezza circa del 93% mentre i modelli che sfruttano un kernel lineare non superano il 92% di accuratezza, circa 91.5%.

In sostanza dopo la normalizzazione le prestazioni dei modelli si avvicinano sensibilmente ai risultati ottenuti dai ricercatori, addirittura sembra che il modello con kernel rbf e addestrato con dataset normalizzato e semplificato porta prestazioni migliori sul dataset rispetto a quello presentato nell'articolo. In ogni caso le considerazioni che possono essere fatte da questi risultati sono essenzialmente due: il primo è che il modello support vector machine con kernel rbf performa leggermente meglio rispetto al modello lineare sebbene quest'ultimo sia comunque relativamente efficiente; il secondo è che la differenza tra l'utilizzo del dataset completo e quello semplificato è minima. Questo comporta che è possibile non considerare le colonne altamente correlate senza perdere prestazioni e che quindi il modello può basarsi su un numero di features inferiore.

Principal Components Analysis

Considerando che le feature nel dataset sono tra loro molte correlate l'idea che ho voluto portare è quella di effettuare una analisi delle componenti principali (PCA) per ridurre il dataset e successivamente eseguire la fase di addestramento. In questo modo si dovrebbe semplificare il training perché si lavorerebbe su dimensioni inferiori sperando di mantenere invariate le performance. Il modello sarebbe quindi più leggero. L'analisi delle componenti è stata effettuata sia sul dataset originale che sul dataset semplificato per poter confrontare poi i risultati tra loro e rispetto ai risultati appena mostrati. Entrambi i dataset sono stati normalizzati riprendendo così le prestazioni dei modelli precedente addestrati.

L'analisi è stata effettuata in modo da ridurre le dimensioni del dataset a solo due features. Questa scelta non è stata dettata per motivi di prestazioni ma per poter poi visualizzare in un piano la distribuzione delle istanze per le varie classi. In questo modo sarebbe possibile vedere come le varie classi sono distribuite anche tra di loro, se in pratica è possibile ottenere un classificatore SVM efficace anche con così poche features. Ottenere risultati migliori potrebbe risultare poco realistico, più probabile l'obiettivo è raggiungere una buona percentuale di accuratezza che potrebbe essere superiore all'80%. L'addestramento è stato poi effettuato sia per i modelli con kernel rbf che con kernel lineare.

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	1886	6	0	0	1	94	40
BARBUNYA	11	726	1	497	14	73	0
BOMBAY	0	0	522	0	0	0	0
CALI	2	205	0	1376	34	13	0
HOROZ	0	1	0	33	1831	47	16
SIRA	42	11	0	1	60	2284	238
DERMASON	55	0	0	0	12	236	3243

Tabella 10 - matrice di confusione modello support vector machine con kernel rbf su dataset ottenuto da pca

Accuratezza: 87,19% - Precisione: 87% - F1: 87%

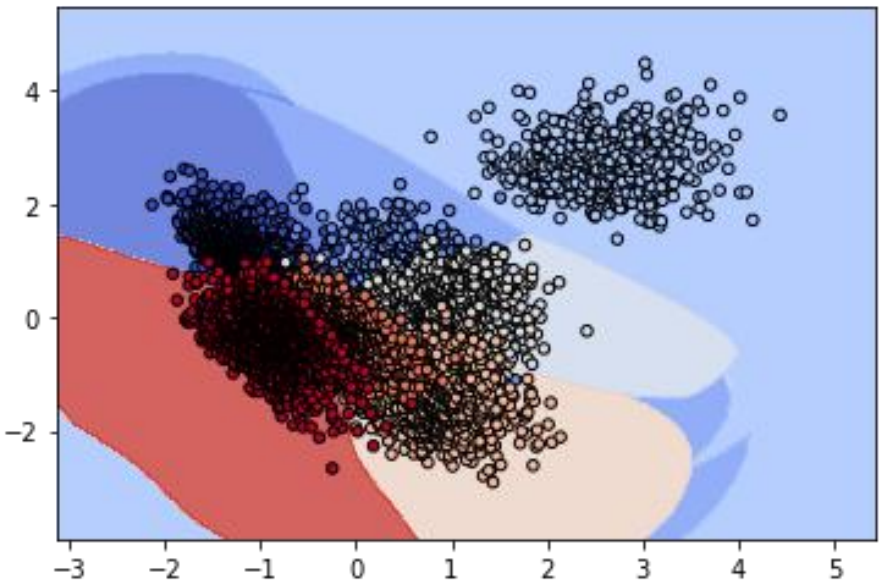

Figura 2 - grafico modello svm con kernel rbf

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	1914	0	0	0	1	82	30
BARBUNYA	86	62	2	907	43	222	0
BOMBAY	1	0	521	0	0	0	0
CALI	2	1	0	1402	171	54	0
HOROZ	0	0	0	18	1850	28	32
SIRA	51	0	0	0	118	2004	463
DERMASON	76	0	0	0	8	105	3357

Tabella 11 - matrice di confusione modello support vector machine con kernel lineare su dataset ottenuto da pca

Accuratezza: 81,62% - Precisione: 84% - F1: 78%

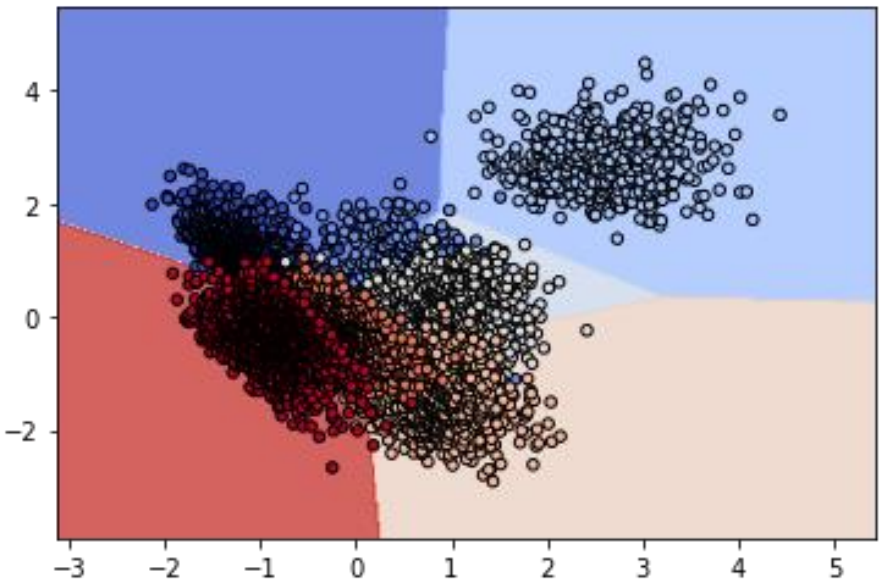

Figura 3 - grafico modello svm con kernel lineare

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	1864	5	0	0	1	105	52
BARBUNYA	7	548	1	633	56	77	0
BOMBAY	0	0	522	0	0	0	0
CALI	2	221	0	1348	46	13	0
HOROZ	0	12	0	44	1796	58	18
SIRA	39	14	0	1	95	2254	233
DERMASON	61	0	0	0	24	226	3235

Tabella 12 - matrice di confusione modello support vector machine con kernel rbf su dataset ottenuto da pca dal dataset semplificato

Accuratezza: 84,98% - Precisione: 85% - F1: 85%

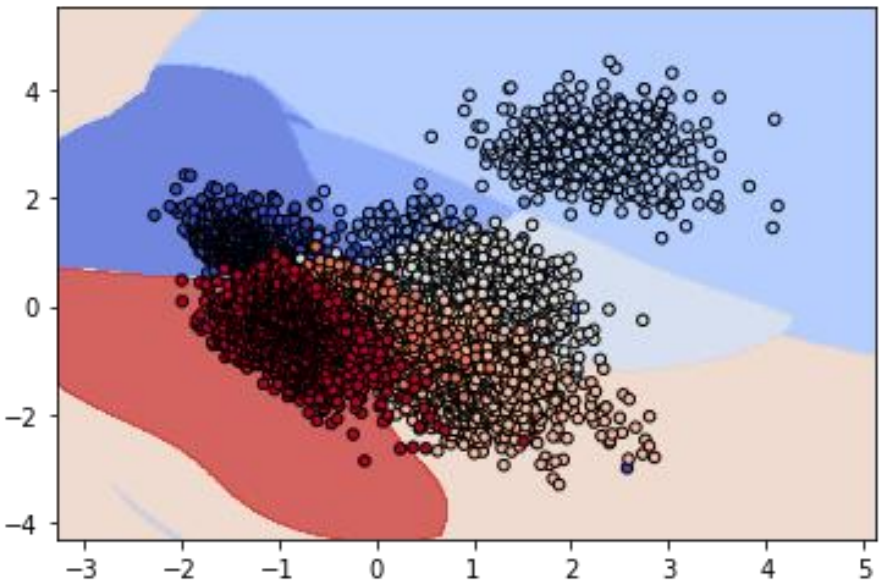

Figura 4 - grafico modello svm con kernel rbf da dataset semplificato

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	1907	0	0	0	1	81	38
BARBUNYA	50	0	1	964	115	191	1
BOMBAY	1	0	521	0	0	0	0
CALI	4	0	1	1359	219	47	0
HOROZ	0	0	0	26	1834	34	34
SIRA	72	0	0	0	170	1951	443
DERMASON	102	0	0	0	18	113	3313

Tabella 13 - matrice di confusione modello support vector machine con kernel lineare su dataset ottenuto da pca dal dataset semplificato

Accuratezza: 79.97% - Precisione: 73% - F1: 76%

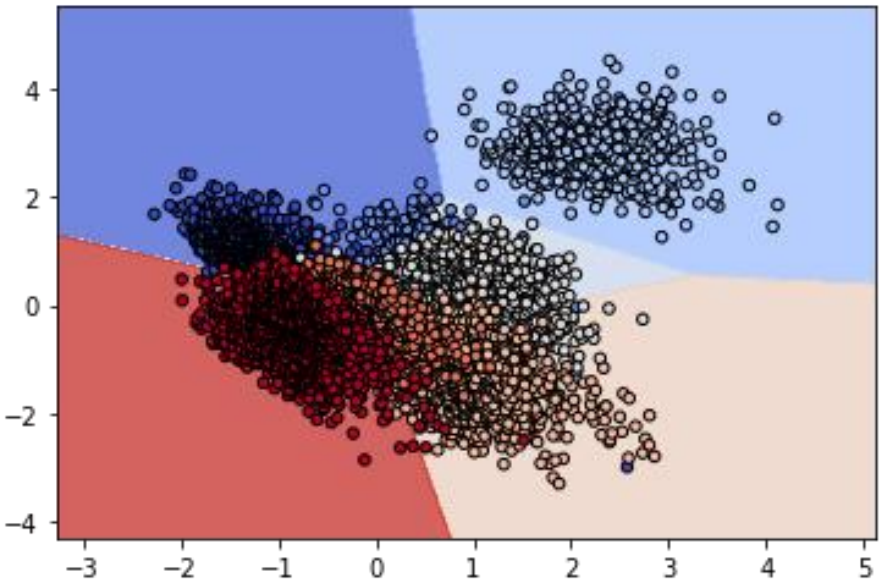

Figura 5 - grafico modello svm con kernel lineare da dataset semplificato

I modelli svm con kernel rbf hanno un'accuratezza migliore rispetto ai modelli che sfruttano un kernel lineare. Infatti i primi si aggirano tra l'85% e l'87% di accuratezza mentre i modelli che usano kernel lineare si aggirano tra l'80% e l'82%. In entrambi i casi i modelli basati sul dataset semplificato hanno prestazioni leggermente peggiori.

I modelli addestrati dopo la pca hanno delle buone prestazioni ma che non equivalgono con quelle precedentemente analizzate. Innanzitutto lavorare sul dataset completo porta a risultati migliori, meglio se viene utilizzato il kernel rbf per il classificatore svm. Il classificatore lineare presenta delle difficoltà date dal fatto che è difficile separare linearmente le classi. Infatti come si può notare dalle matrici di confusione la classe BARBUNYA viene spesso classificata come CALI, avendo così una misclassificazione più frequente in questa classe. Il caso più lampante è il modello svm lineare con dataset semplificato: in questo modello si può vedere come nessuna delle istanze viene riconosciuta come BARBUNYA e che quindi la sua regione è nulla. Se negli altri tre casi il modello aveva una percentuale di errore maggiore del 50% per quella classe, ovvero c'era un numero di istanze classificate come CALI rispetto a quelle classificate correttamente, in quest'ultimo non c'è modo per alcuna istanza di poter ottenere come risultato la classe BARBUNYA. Pertanto il classificatore lineare non è indicato per avere un buon modello, sebbene per le restanti classi la classificazione sia comunque accettabile. Questo comportamento si spiega facilmente in quanto le due classi si sovrappongono del piano avente come misure dei due assi le features ottenute dalla pca. Sebbene la pca ha permesso di poter ottenere un grafico visualizzabile del dataset, ha d'altro canto perso una certa quantità di informazioni che non ha più permesso di poter discriminare le due classi.

Questo problema non è irrisolvibile, anzi, nel caso in cui si aumentassero il numero di features dalla fase di pca, magari a 5 o 4, anche fino a 3 ipoteticamente, è più probabile che riconoscere le due classi sia nettamente più semplice e di conseguenza si otterrebbero prestazioni migliori. Ciononostante questi esperimenti hanno permesso di evincere come due classi all'interno del dataset sono molto simili come caratteristiche e che quindi un modello potrebbe fare fatica a discriminarle.

RANDOM FOREST

Un altro approccio possibile proposto dai ricercatori è l'albero decisionale, modello abbastanza semplice da addestrare ma risulta anche molto intuitivo da interpretare. Capire quali sono le caratteristiche che permettono di discriminare le classi è quasi diretto in quanto sono le condizioni if-else che permettono di

percorrere un ramo dell'albero piuttosto che un altro. Il dataset e le proprietà delle features si apprestano particolarmente bene alla tipologia del modello, motivo per il quale anche i ricercatori hanno deciso di adoperare tale metodo. L'albero decisionale è stato addestrato con massimo numero di partizioni pari a 16 e il criterio di partizione usato è l'indice Gini. Di seguito la matrice di confusione del loro modello.

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	1877	7	0	0	0	81	62
BARBUNYA	12	904	1	352	3	50	0
BOMBAY	0	1	517	3	1	0	0
CALI	1	140	0	1455	24	10	0
HOROZ	1	2	0	102	1709	99	15
SIRA	45	2	0	28	7	2296	258
DERMASON	73	0	0	0	1	263	3209

Tabella 14 - matrice di confusione del modello albero decisionale proposto nell'articolo

Le prestazioni dell'albero decisionale sono buone ma è il peggior modello proposto dagli autori. Il modello ha un'accuratezza del 87,92% ed error rate al 12,08% mentre la precisione di assesta al 89,19% e F1-score si aggira al 88,29%. Di seguito l'albero ottenuto e mostrato nell'articolo.

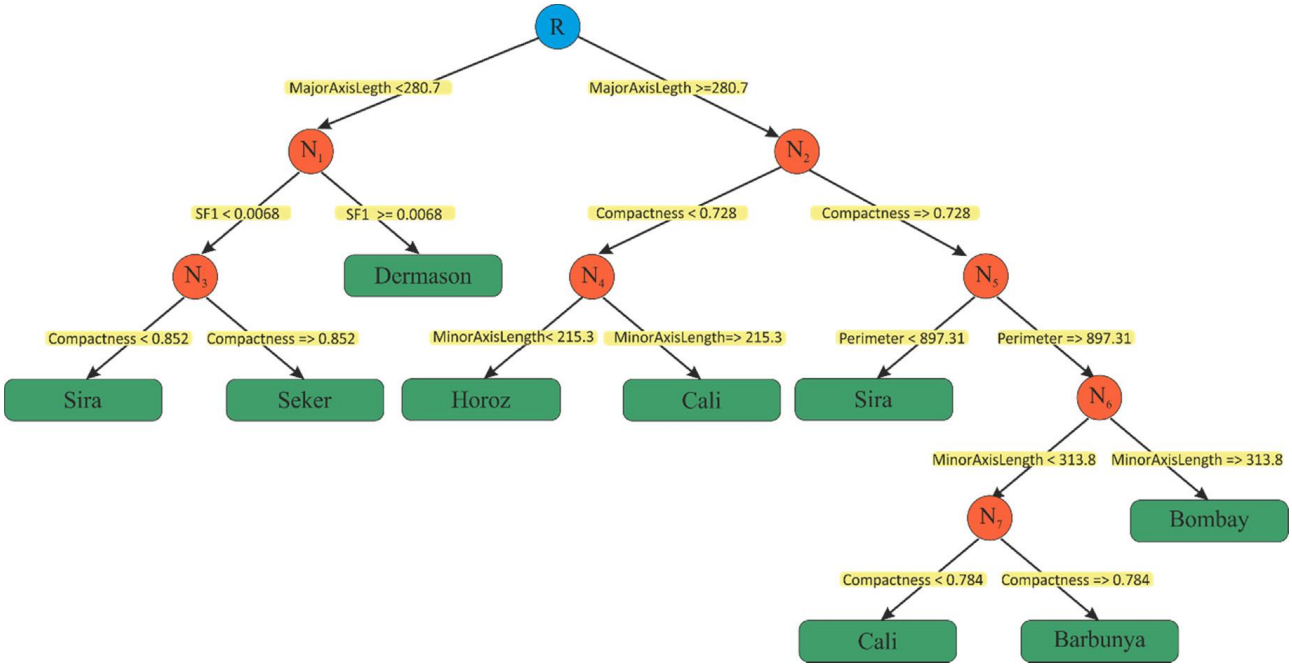

Figura 6 - struttura albero decisionale mostrato nell'articolo

Perciò ho provato abbastanza velocemente a costruire un mio modello di albero decisionale per controllare i possibili risultati, utilizzando il dataset originale e completo. Ho separato il dataset in un training set e in un test set per poi verificare le prestazioni del modello. Di seguito vengono mostrati i risultati ottenuti.

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	228	0	85	0	5	4	8
BARBUNYA	0	129	1	0	0	0	0
BOMBAY	29	0	369	0	7	1	2
CALI	0	0	0	786	0	19	82
HOROZ	0	0	27	6	439	0	10
SIRA	2	0	0	12	0	471	22
DERMASON	0	0	4	56	10	11	578

Tabella 15 - matrice di confusione del modello albero decisionale

Accuratezza: 88.93% - Precisione: 90% - F1: 89%

Feature	Importance
ShapeFactor1	0.281375
Compactness	0.262525
MajorAxisLength	0.223023
Perimeter	0.170017
MinorAxisLength	0.031820
Roundness	0.031240

Tabella 16 - feature importance ricavati dall'albero decisionale

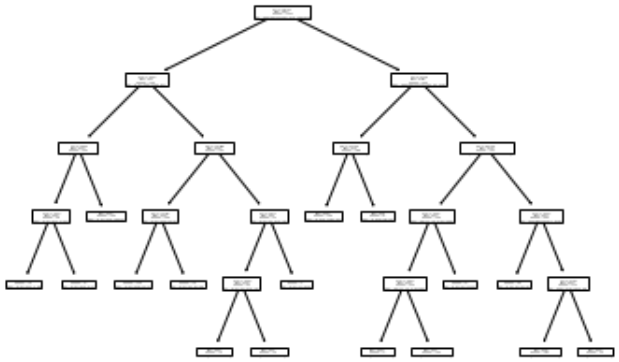

Figura 7 - struttura albero decisionale costruito

Come si può notare le prestazioni con questa semplice prova, dove è stato impostato come livello di profondità massimo pari a 5 e numero massimo di foglie pari a 16, le prestazioni sono buone e si avvicinano e superano le prestazioni del modello proposto dagli autori dell'articolo raggiungendo un'accuratezza dell'89%.

Un particolare che può essere messo in risalto dagli alberi di decisione è la feature importance, ovvero un indicatore che rappresenta l'importanza di una particolare feature nel discriminare una classe nel modello. Sopra infatti è stata riportata una piccola tabella in relazione all'albero decisionale costruito. Questo concetto molto interessante è diventato poi il punto di partenza del metodo proposto ovvero sull'utilizzo delle random forest. La tabella delle feature importances varia da albero ad albero, a seconda di quali caratteristiche sono utilizzate per riconoscere le classi. Quindi sarebbe possibile con una random forest costruire tutta una serie di alberi in modo da poter da una parte ottenere performance migliori con questo modello e dall'altra parte capire quali sono in media le features che sono più significative e che permettono di classificare con maggior facilità le istanze.

Il tentativo effettuato con il metodo delle random forest ha previsto anche in questo caso della creazione di un dataset di training e uno di test a partire dal dataset originale. In questo modo sarà per noi possibile giudicare la bontà del modello addestrato. Non sono stati impostati particolari valori nei parametri disponibili nel metodo. Di seguito sono riportati i risultati, comprensivi di performance e matrice di confusione rispetto al dataset di test.

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	296	0	19	0	1	6	8
BARBUNYA	0	130	0	0	0	0	0
BOMBAY	13	0	382	0	8	2	3
CALI	0	0	0	812	0	18	57

HOROZ	1	0	5	7	461	0	8
SIRA	2	0	0	9	0	484	12
DERMASON	4	0	0	65	8	10	572

Tabella 17 - matrice di confusione modello random forest

Accuratezza: 92.18% - Precisione: 94% - F1: 93%

Come già mostrato con il modello dell'albero decisionale, è possibile poi evidenziare l'importanza di ciascuna feature in relazione a ciascun albero che compone il modello della foresta appena costruito. Queste importances sono mostrati nel grafico seguente. Per avere maggior comprensione del grafico è utile considerare la numerazione delle feature è la seguente:

0. *Area*
1. *AspectRation*
2. *Compactness*
3. *ConvexArea*
4. *Eccentricity*
5. *EquivDiameter*
6. *Extent*
7. *MajorAxisLength*
8. *MinorAxisLength*
9. *Perimeter*
10. *ShapeFactor1*
11. *ShapeFactor2*
12. *ShapeFactor3*
13. *ShapeFactor4*
14. *Solidity*
15. *Roundness*

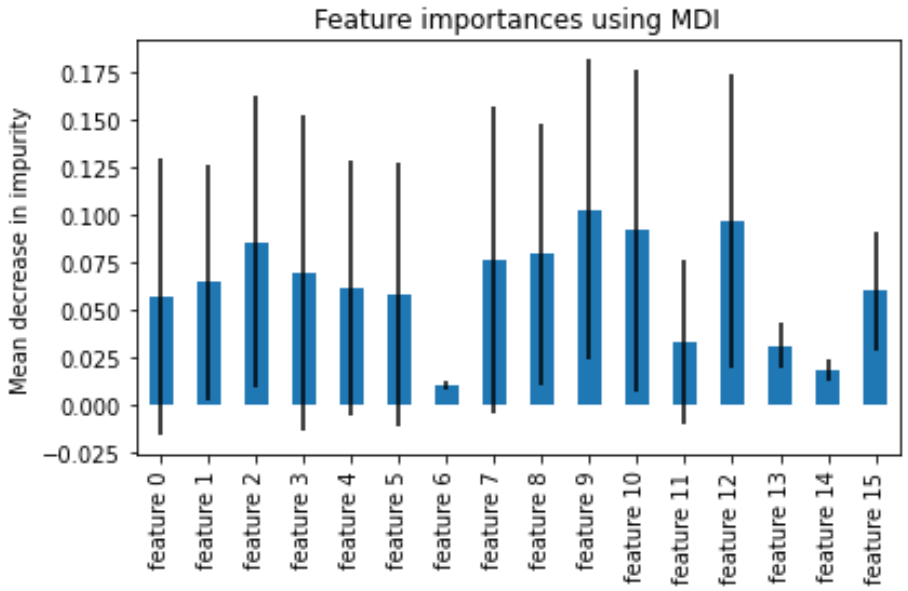

Figura 8 - feature importances ricavati dal modello random forest

Come è possibile notare dal grafico, non esiste una particolare feature che ha una sostanziale importanza rispetto alle altre. Quasi tutte hanno all'incirca lo stesso livello di importanza (barra blu) mentre hanno una varianza notevole (barra sottile nera). Questo può risultare conseguenza della caratteristica del dataset. Il fatto che molte delle features che compongono il dataset sono fortemente correlate fra loro indica come è possibile utilizzare una feature piuttosto che un'altra per poter classificare in maniera più o meno efficace le istanze. La grande varianza indica inoltre che essendo molto correlate fra loro non è necessario avere considerazione di tutte le features per poter effettuare la classificazione in maniera corretta. Quindi plausibilmente un albero decisionale non ha bisogno di tutte le proprietà per la classificazione ma soltanto di alcune. Se non è possibile ottenere quali sono le features più importanti, è però possibile notare come alcune features sono poco utili per discriminare la classe all'interno del modello. In particolar modo queste features sono Extent e Solidity alle quali si potrebbe anche aggiungere ShapeFactor4. Quindi ho provato a togliere le due feature meno importanti e ripetere l'esperimento. I risultati ottenuti sono mostrati di seguito e ricalcano più o meno i risultati precedentemente ottenuti.

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	291	0	23	0	1	6	9
BARBUNYA	0	130	0	0	0	0	0
BOMBAY	17	0	378	0	8	2	3
CALI	0	0	0	813	0	15	59
HOROZ	2	0	5	5	462	0	8
SIRA	2	0	0	8	0	484	13
DERMASON	3	0	0	72	7	11	566

Tabella 18 - matrice di confusione nuovo modello random forest togliendo features poco significative

Accuratezza: 91.80% - Precisione: 93% - F1: 93%

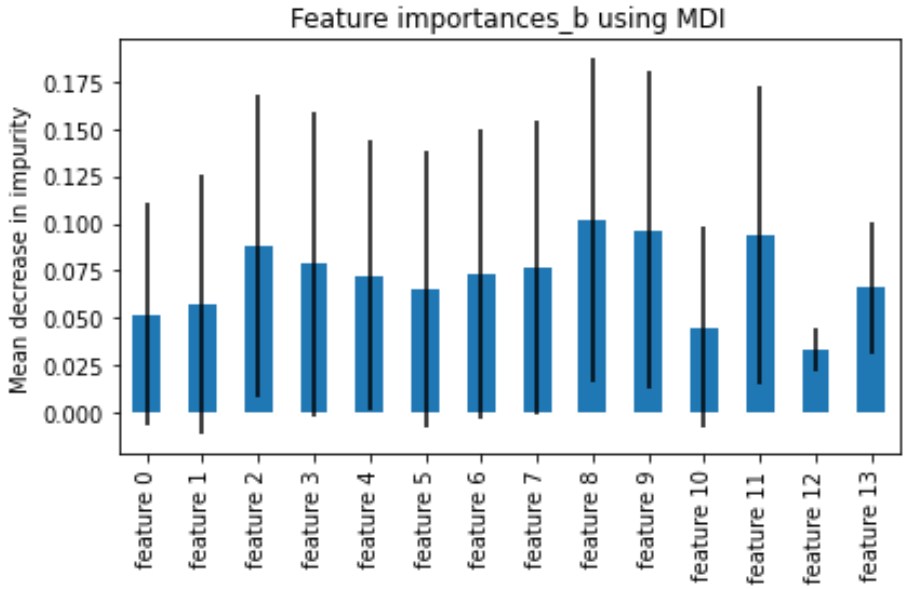

Figura 9 - feature importances ricavati dal nuovo modello random forest

Le prestazioni sono rimaste pressoché invariate, sono leggermente inferiori ma in modo poco significativo. Il grafico delle feature importances seguono il grafico precedente.

Guardando i risultati ottenuti si può evidenziare come la classe BARBUNYA è la classe meglio riconoscibile ottenendo una precisione massima circoscritta alla singola classe. In generale il metodo delle random forest comporta un aumento delle prestazioni e pertanto è un miglioramento delle performances rispetto al modello proposto dagli autori.

NEURAL NETWORK

L'ultima tipologia di modelli che ho approfondito per il dataset in questione è quello della rete neurale. Modello utilizzato anche dagli autori del paper di riferimento con il loro modello Multi-Layer Perceptron. In definitiva hanno creato una struttura a quattro layer: il primo layer è quello di ingresso che riceve le 16 features di una istanza da classificare sommato a un ingresso per il bias; il secondo e il terzo sono layer intermedi, rispettivamente di 12 e 3 nodi; l'ultimo layer, quello di output, rappresenta l'uscita che permette la classificazione dell'istanza. Nella fase di tuning hanno raggiunto un modello ideale impostando come learning rate pari a 0.3 e come funzione di attivazione la funzione sigmoide. Per l'addestramento è stato impostato come numero massimo di epoche 500. Di seguito viene mostrata l'architettura progettata dagli autori seguita dalla matrice di confusione in riferimento all'intero dataset.

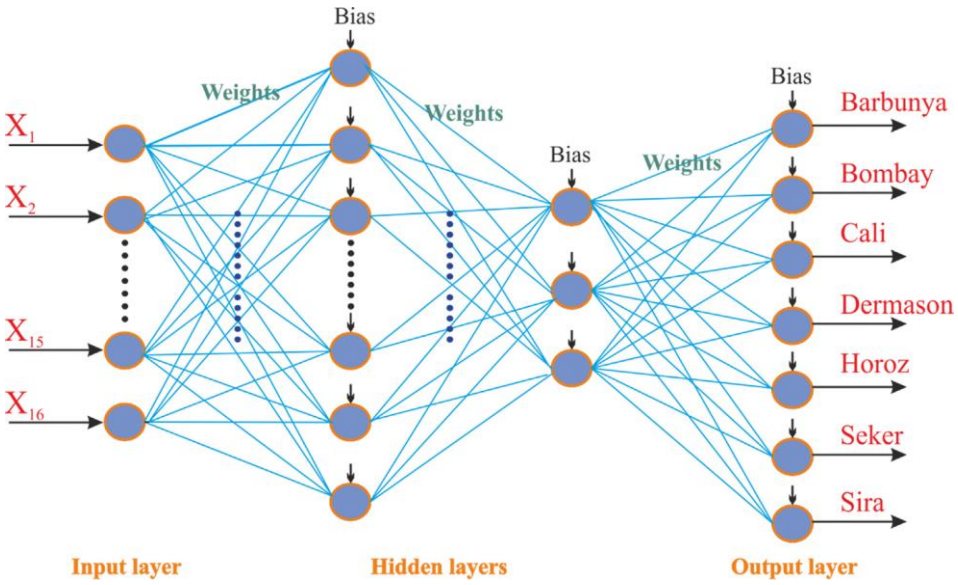

Figura 10 - architettura rete neurale proposta nell'articolo

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	1899	14	0	0	1	64	49
BARBUNYA	10	1184	4	87	3	32	2
BOMBAY	0	1	518	3	0	0	0
CALI	3	44	5	1530	34	14	0
HOROZ	0	5	0	29	1829	49	16
SIRA	26	9	0	4	40	2232	325
DERMASON	52	1	0	0	2	198	3293

Tabella 19 - matrice di confusione modello rete neurale proposto nell'articolo

Le prestazioni della rete neurale sono particolarmente buone, infatti il modello ha un'accuratezza del 91.73% ed error rate al 8.27% mentre la precisione di assesta al 93.11% e F1-score si aggira al 92.88%.

L'architettura utilizzata rispecchia lo stato dell'arte delle reti neurali quando l'articolo è stato pubblicato. Pertanto il mio obiettivo era quello di progettare una rete neurale che possa superare le prestazione del modello appena visto utilizzando altre tecniche che in questi anni sono emerse e/o andate di moda. La struttura della rete è rimasta pressoché la stessa, i cambiamenti più significativi si possono notare nel secondo livello nascosto che vede il numero di nodi aumentare da 3 a 6 e un livello che precede il livello di input che prevede un'operazione di normalizzazione batch. In questo modo le istanze potranno essere valutate in modo migliore e si evita situazioni in cui la rete non riesce a trovare un modello che non polarizzi su una sola classe. Senza infatti questa fase o senza questo layer la rete neurale fa fatica a ottenere un modello decente ma semplicemente si ritrova a classificare tutte le istanze verso un'unica classe ottenendo perciò delle prestazioni imbarazzanti. Il layer può essere omesso qualora si decidesse di effettuare la procedura di normalizzazione del dataset precedentemente, così come è stato fatto per il metodo delle support vector machine.

Le novità introdotte nel mio modello architetturale sono passaggio dalla funzione sigmoide alla ReLu come funzione di attivazione e l'utilizzo di una softmax come funzione di attivazione nel livello di uscita. L'utilizzo di una softmax in uscita ha comportato la trasformazione delle etichette delle classi in forma di array binario tramite il metodo to_categorical. Inoltre sono stati fatti tentativi con l'introduzione di livelli intermedi di dropout con un rate pari al 0.2. Le varie tipologie di modelli son stati testati e adattati anche per verificare le prestazioni addestrando la rete con il dataset semplificato. In generale le aspettative migliori sono riposte nel modello originale proposto mentre le varianti saranno valutate per poter vedere quali differenze ricorrono.

Per quanto riguarda l'ottimizzazione si è deciso di adoperare l'ottimizzatore Adam mentre per la fase di training è stato impostato 40 come numero di epoche con dimensione dei batch pari a 50. Il learning rate era impostato inizialmente a 0.03 ma, osservando iniziali test, si è visto che non produceva prestazioni interessanti e pertanto è stato deciso di seguire il rate utilizzato dai ricercatori, ovvero un learning rate pari a 0.007. Inoltre si è tenuto conto anche della validazione nell'addestramento impostando un validation split al 10%.

Un elemento che è stato tenuto in considerazione per la fase di addestramento delle reti neurale è il numero di istanze per classe. Questo numero è molto variabile e infatti si passa dalla classe BOMBAY con appena 522 istanze mentre altre classi come SIRA e DERMASON hanno un numero nettamente maggiore, rispettivamente 2636 e 3546. Quindi ogni classe è stata pesata per l'addestramento in modo da evitare che la rete apprenda maggiormente una classe piuttosto che un'altra.

Tutte queste caratteristiche sono inoltre state utilizzate per adattare l'architettura della rete neurale per poter utilizzare il dataset semplificato con lo scopo di poter commentare la differenza di prestazioni. In questo caso l'unica differenza sostanziale risiede nel livello di ingresso che è stato impostato per accettare il numero di parametri corretto ovvero accettare le 11 features che compongono il dataset semplificato piuttosto che le 16 del dataset originale.

Di seguito sono riportati i risultati delle reti neurali addestrate con l'architettura senza layer di dropout. Entrambe le versioni delle reti neurali, ovvero per accettare il dataset originale e quello semplificato, hanno ottenuto delle ottime prestazioni, superando il 92% di accuratezza e migliorando seppur di poco le performance della rete dei ricercatori. In particolare l'addestramento è particolarmente rapido e già dopo un numero relativamente basso di epoche si raggiunge un buon livello di accuratezza, come si può notare dai grafici che seguono le matrici di confusione.

	SEKER	**BARBUNYA**	**BOMBAY**	**CALI**	**HOROZ**	**SIRA**	**DERMASON**
SEKER	1945	8	0	0	0	52	22
BARBUNYA	10	1165	1	106	4	36	0
BOMBAY	0	0	522	0	0	0	0
CALI	3	18	0	1553	23	33	0
HOROZ	1	2	0	28	1821	54	22
SIRA	37	6	0	2	25	2300	266
DERMASON	72	0	0	0	1	163	3310

Tabella 20 - matrice di confusione modello rete neurale

Accuratezza: 92.69%; Precisione: 94.11% - F1: 93.76%, accuratezza nel test set: 92.80%

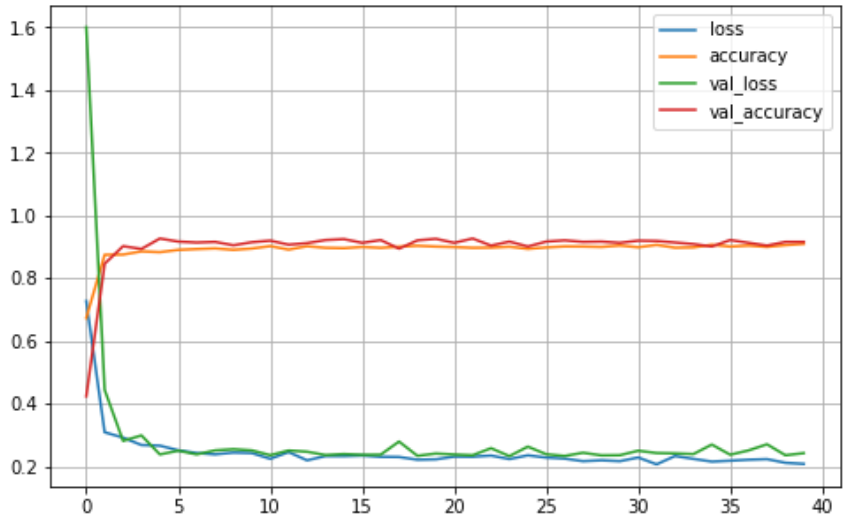

Figura 11 - grafico dell'andamento di accuracy e loss durante l'addestramento della rete neurale

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	1954	8	0	0	0	49	16
BARBUNYA	9	1222	3	61	2	25	0
BOMBAY	0	0	522	0	0	0	0
CALI	3	43	34	1523	16	11	0
HOROZ	0	6	0	49	1810	46	17
SIRA	40	8	0	8	26	2398	156
DERMASON	77	0	0	0	5	319	3145

Tabella 21 - matrice di confusione modello rete neurale addestrato con dataset semplificato

Accuratezza: 92.38%; Precisione: 92.98% - F1: 93.28%, accuratezza nel test set: 92.89%

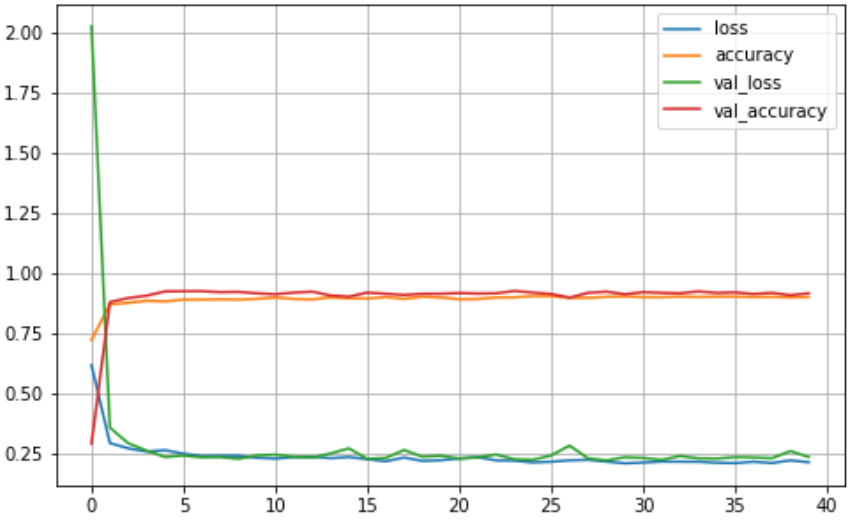

Figura 12 - grafico dell'andamento di accuracy e loss durante l'addestramento della rete neurale costruita per il dataset semplificato

Per quanto riguarda la differenza tra i due dataset, le due reti neurali non hanno evidenziato particolari divergenze ma hanno mostrato come la rimozione di colonne che sono altamente correlate non compromettono le prestazioni del modello e che quindi la rete è capace di classificare il più delle volte in modo corretto le istanze. Anche considerando le singole classi nelle matrici di confusione si può notare che la misclassificazione è del tutto simile per le due reti e perciò non c'è una sostanziale differenza nemmeno

per il riconoscimento di una classe in particolare. A riprova di questo vi è anche il pressoché identico livello di performance che caratterizzano le due reti.

Perciò è più indicativo confrontare differenze tra reti con diverse architetture e verificare nel nostro caso se l'introduzione dei dropout comporta un cambiamento sostanziale.

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	1904	28	0	0	0	70	25
BARBUNYA	8	1192	0	94	1	27	0
BOMBAY	0	0	520	2	0	0	0
CALI	2	37	0	1559	16	16	0
HOROZ	0	2	0	84	1773	51	18
SIRA	33	9	0	15	26	2366	187
DERMASON	64	2	0	0	2	317	3161

Tabella 22 - matrice di confusione modello rete neurale con livelli di dropout

Accuratezza: 91.65%; Precisione: 93.04% - F1: 92.90%, accuratezza nel test set: 91.95%

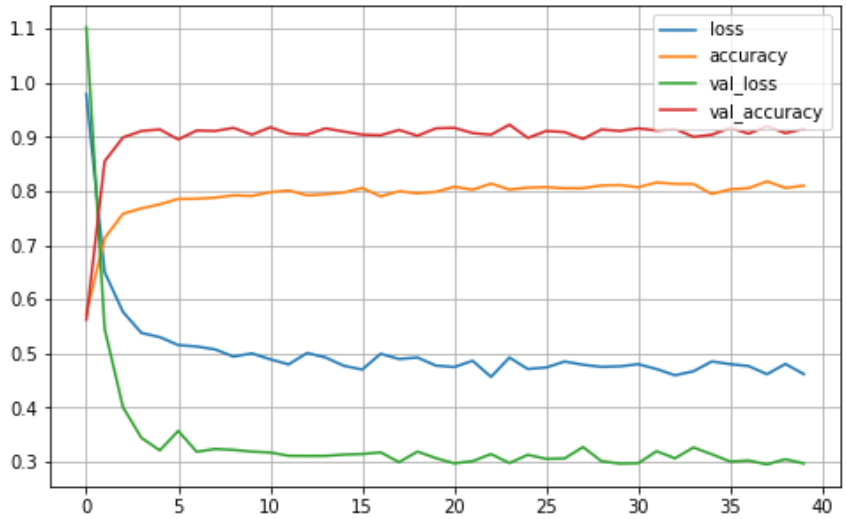

Figura 13 - grafico dell'andamento di accuracy e loss durante l'addestramento della rete neurale con livelli dropout

	SEKER	BARBUNYA	BOMBAY	CALI	HOROZ	SIRA	DERMASON
SEKER	1903	16	0	0	0	58	50
BARBUNYA	7	1195	4	58	5	53	0
BOMBAY	0	0	521	1	0	0	0
CALI	3	59	6	1480	39	43	0
HOROZ	0	2	0	23	1839	44	20
SIRA	17	4	0	1	39	2313	262
DERMASON	39	0	0	0	3	199	3305

Tabella 23 - matrice di confusione modello rete neurale con livelli di dropout addestrato con dataset semplificato

Accuratezza: 92.25%; Precisione: 93.55% - F1: 93.27%, accuratezza nel test set: 92.27%

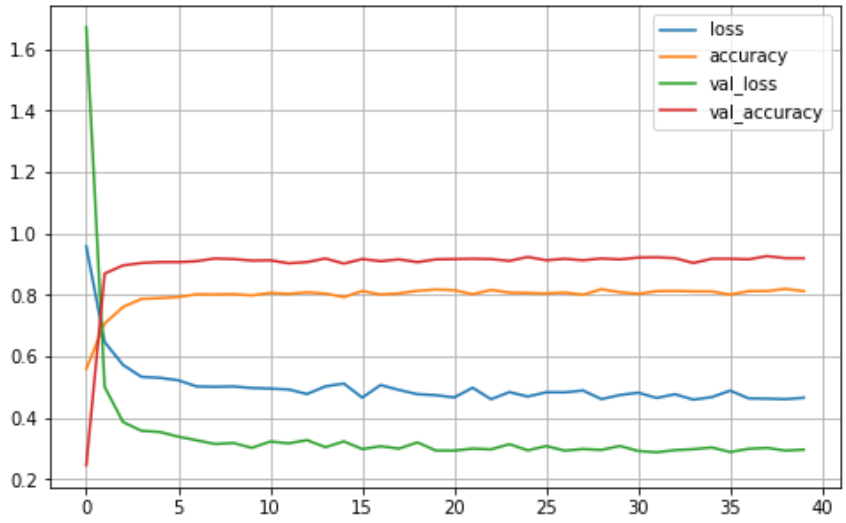

Figura 14 - grafico dell'andamento di accuracy e loss durante l'addestramento della rete neurale con livelli dropout costruita per il dataset semplificato

Come nel caso precedente, anche inserendo i layer di dropout le differenze tra la rete che accetta il dataset originale e quella che accetta quella semplificata sono praticamente trascurabili. Le differenze di prestazione non sono sostanziali, forse si potrebbe notare una migliore efficienza per la rete neurale addestrata con il dataset semplificato, ma in linea di massima entrambe le reti addestrate dimostrano delle buone performance che superano le prestazioni del modello proposto dai ricercatori. Uno dei pochi aspetti che si possono notare con l'utilizzo o meno dei dropout è il valore di accuracy durante l'addestramento. Nella rete dei dropout questo si assesta a un 80% circa ma questo fa riferimento all'accuratezza rispetto al training set usato durante l'epoca mentre l'accuratezza riferita al validation set è molto alta e corrisponde ai valori di efficienza misurati successivamente con il test set. Perciò questa differenza di metodo risulta non significativa rispetto alle prestazioni del modello.

In generale ogni variante della rete neurale proposta ha portato dei buoni risultati, migliorando le prestazioni raggiunte dai ricercatori, mostrando un ottima abilità di riconoscere la classe corretta per praticamente tutte le classi e raggiungendo gli obiettivi prefissati per questo modello.

CONSIDERAZIONI FINALI

Commentando i risultati ottenuti, si può vedere come i modelli proposti hanno portato dei buoni risultati producendo un classificatore che fosse in grado di riconoscere in modo corretto la tipologia di semi trattati. Le performance ottenute per ciascun modello, considerando le versioni migliori, sono soddisfacenti superando il 90% di accuratezza. Sebbene sia buono come valore prestazionale è indicativo come questo non superi il 95% e nemmeno si avvicini a valori vicini al 99%. I motivi di tale comportamento possono essere molteplici ma in ogni caso possono essere riconducibili alla natura del dataset trattato.

Innanzitutto abbiamo visto come molte delle feature siano tra loro fortemente correlate. Questo di per se non comporta necessariamente una riduzione dell'efficienza dei classificatori ma indica come trattare solo caratteristiche di forma e nello specifico caratteristiche geometriche sia alquanto limitativo. Nel caso ci fosse all'interno del dataset qualche caratteristica come il colore probabilmente la classificazione in alcuni contesti sarebbe stata nettamente più semplice, sebbene non determinante visto alcune tipologie di semi condividono il colore. Inoltre è necessario specificare come questa scelta riguardo le caratteristiche sia dovuta poi a un'analisi automatizzata utilizzando tecnologie di elaborazione delle immagini, ad esempio sfruttando le librerie openCV. La scelta quindi ricadeva poi sull'implementazione del modello in contesti reali e quindi come scelta può essere giustificata.

In secondo luogo le varie specie di semi non necessariamente sono facilmente distinguibili ma in alcuni casi la similitudine può essere nettamente maggiore tra loro per alcune classi e quindi più difficile ottenere una corretta classificazione, al contrario di altre che sono facilmente distinguibili e hanno un livello di accuratezza specifico per la classe in questione superiore rispetto alle altre. In particolar modo notiamo come la classe BOMBAY sia la classe che ha le migliori prestazioni in quanto questa tipologia sia nettamente diversa rispetto agli altri tipi di fagioli essendo i primi nettamente più grandi. Avendo una caratteristica che si discosta maggiormente rispetto agli altri comporta che questa classe sia quella più facile da distinguere. Al contrario classi come DERMASON e SIRA o anche SEKER, soprattutto in modelli che prevedono una forte diminuzione delle caratteristiche come nel caso dei modelli support vector machine seguite alla pca, mostrano come le classi sia vicine tra loro e in alcuni condividono la regione di appartenenza. Infatti per questo motivo questi modelli hanno subito una diminuzione dell'accuratezza evidenziando come la classe BOMBAY si discosta maggiormente dalle altre che invece in alcuni casi hanno fenomeni di overlapping. Nei modelli migliori questo fenomeno non è catastrofico, infatti i modelli hanno buone prestazioni, ma mostra come difficilmente le prestazioni possano essere nettamente migliori.

Infine per avere conferma di tale tesi ho anche testato il dataset con il tool sklearn che permette l'analisi e l'addestramento di una serie di modelli già predisposti in modo da poter constatare la bontà delle performances. Sono stati concessi 5 minuti di tempo al metodo per completare i training e lo score che è risultato da tale metodo mostra un'accuratezza del 93.09%. Di seguito viene riportata una grafica che mostra i modelli testati e a lato la loro accuratezza.

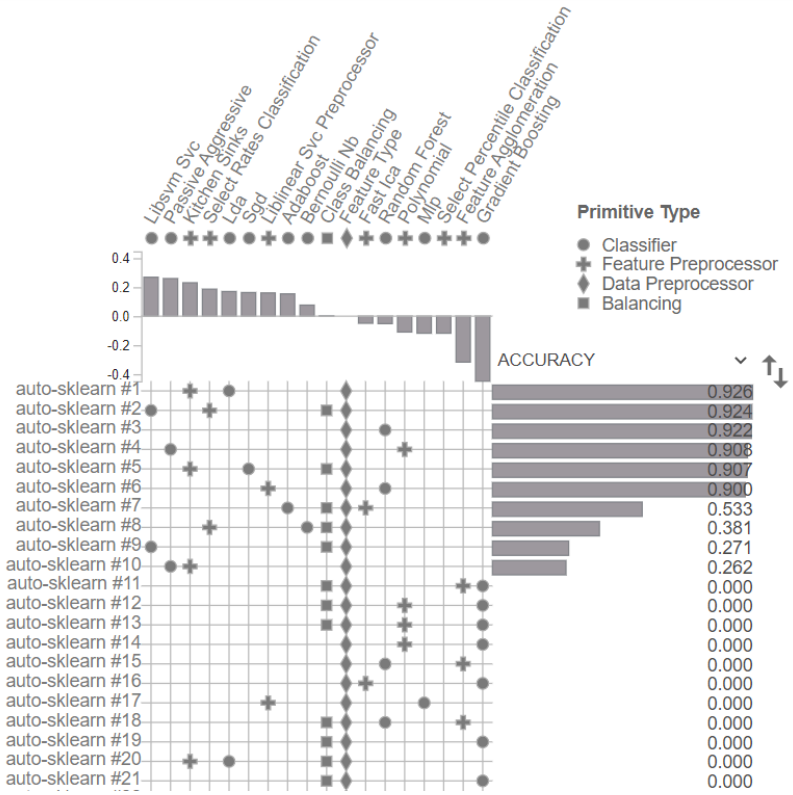

Figura 15 - grafico risultati modelli addestrati con il metodo sklearn

La classifica ordina i modelli dal migliore al peggiore e mostra come i modelli che meglio hanno appreso le differenze tra le classi hanno una accuratezza che si aggira al 93%, in linea con i risultati da noi ottenuti. Questo test è a conferma della nostra tesi ma non è escluso che qualora si ottimizzassero tali modelli con qualche metodo non ancora sperimentato non si possa migliorare di 1-2 punti percentuale, senza ricadere nell'overfitting rispetto al dataset utilizzato.