UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

Machine learning and Data Mining
A.A. 2020-2021
Proff. Ivan Serina
Alfonso Emilio Gerevini

Analisi del dataset UCI ”Seeds”

Alessandro Gaudenzi

Sommario esecutivo

La seguente relazione contiene la descrizione dell'elaborato svolto nell'ambito del corso di Machine Learning e Data Mining.

L'elaborato consiste nell'analisi di un dataset contenente informazioni dimensionali sui semi di grano di tre varietà differenti, nell'addestramento di classificatori di nuove istanze di semi sulla base delle istanze già note e infine nell'ottimizzazione di questi classificatori attraverso il tuning degli iperparametri.

Il linguaggio di programmazione usato per svolgere il lavoro è Python, con il supporto di opportune librerie dedicate all'analisi e alla visualizzazione dei dati, come numpy per l'elaborazione numerica, pandas per svolgere operazioni sui dati efficientemente, matplotlib per visualizzare i dati sottoforma di grafico e sklearn per l'implementazione, l'addestramento e l'ottimizzazione degli algoritmi.

Indice

1 Descrizione dataset

Il dataset esaminato contiene dati dimensionali riguardanti tre varietà di grano: Kama, Rosa e Kanadian. I dati sono stati raccolti dall'istituto di agrofisica dell'università polacca di Lublino, usando una tecnica a raggi X non distruttiva, più economica di altre come lo scanning al microscopio o tecniche al laser.

Il campione è formato da 210 elementi, risulta perfettamente bilanciato contenendo 70 soggetti per ogni varietà e non contiene valori nulli o invalidi.

I parametri geometrici utilizzati come features e associati ad ogni soggetto sono i seguenti: area, perimeter, compactness, length, width, asimmetry e groove.

Nella figura 1 è mostrata la matrice di correlazione tra le feature: possiamo notare che le principali caratteristiche geometriche sono correlate fra loro e l'unica feature realmente indipendente dalle altre è l'asimmetria.

Nelle figure da 2 a 15 viene visualizzato come sono distribuiti i valori delle feature del dataset in base alla classe, per avere un'idea di base della possibile importanza di ciascuna feature nella classificazione e per scrutare la presenza di outliers, che risultano in numero ridotto.

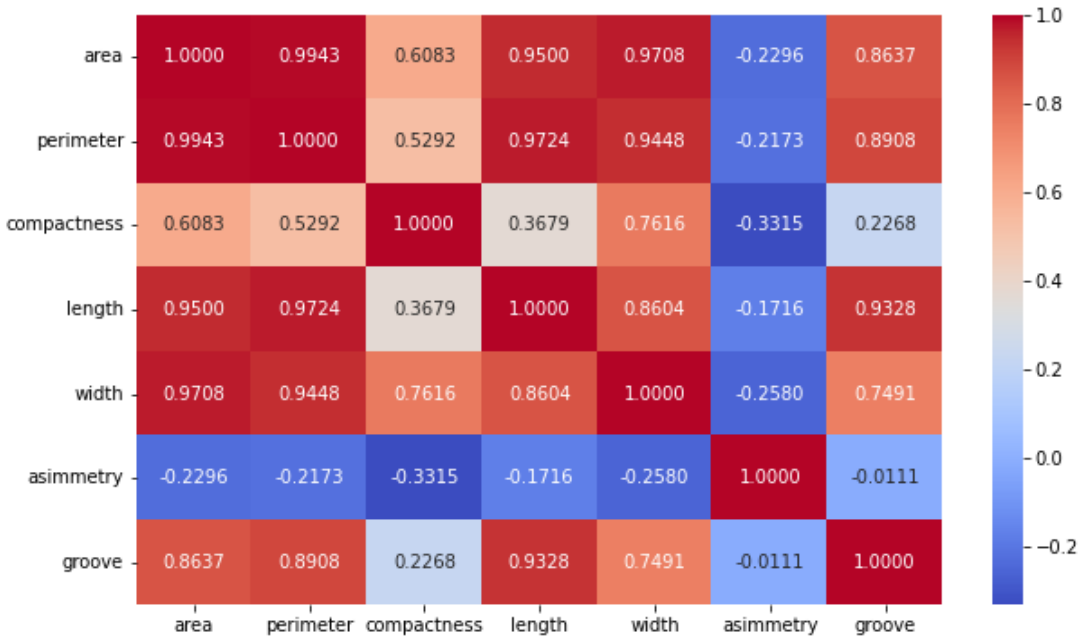

Figura 1: Matrice di correlazione delle feature

1.1 Area

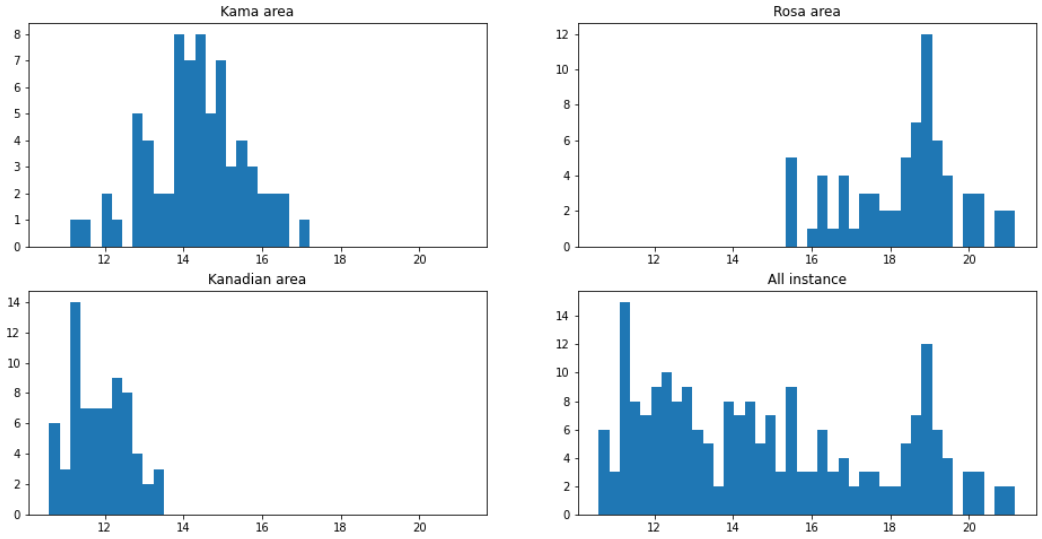

Figura 2: Istogramma della feature area per varietà e totale

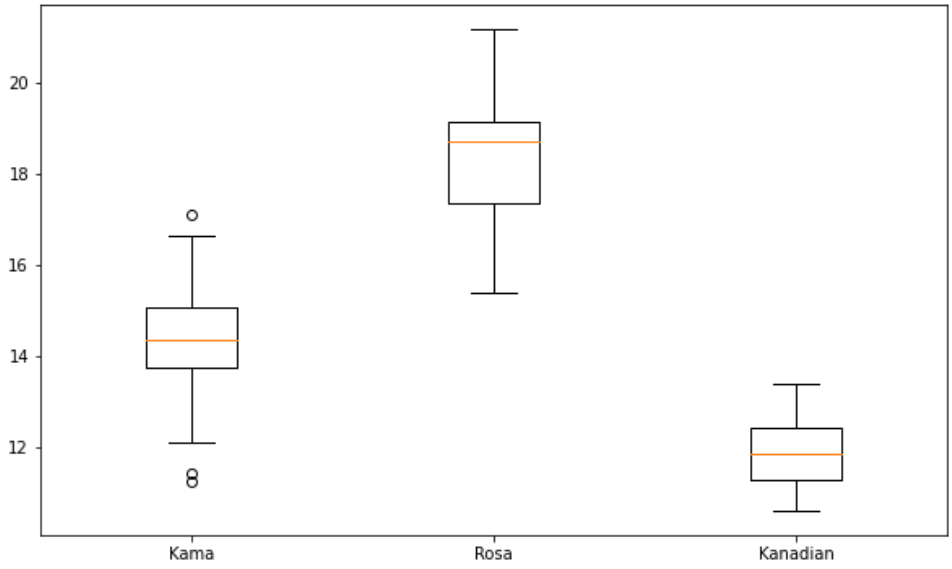

Figura 3: Boxplot della feature area per varietà

1.2 Perimeter

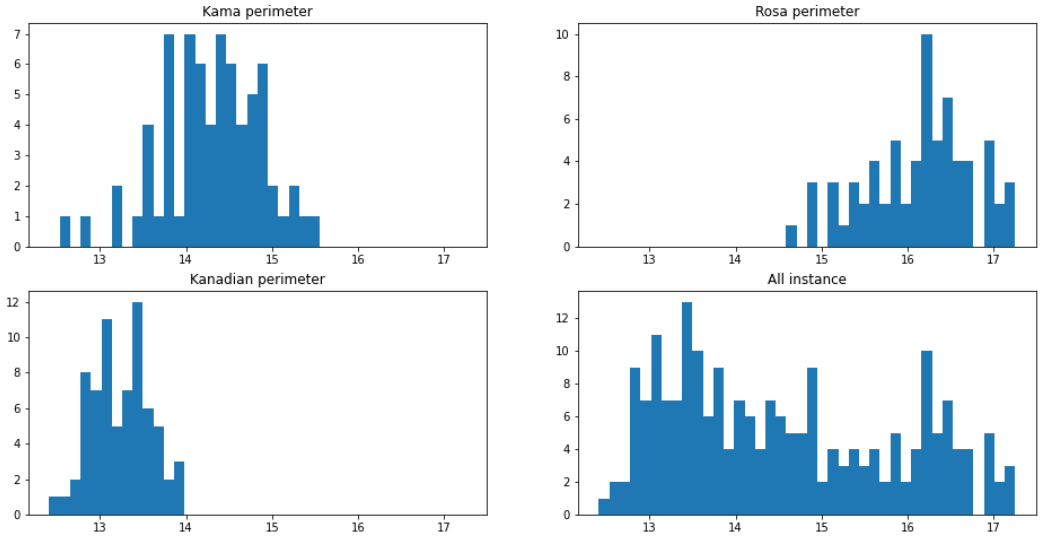

Figura 4: Istogramma della feature perimeter per varietà e totale

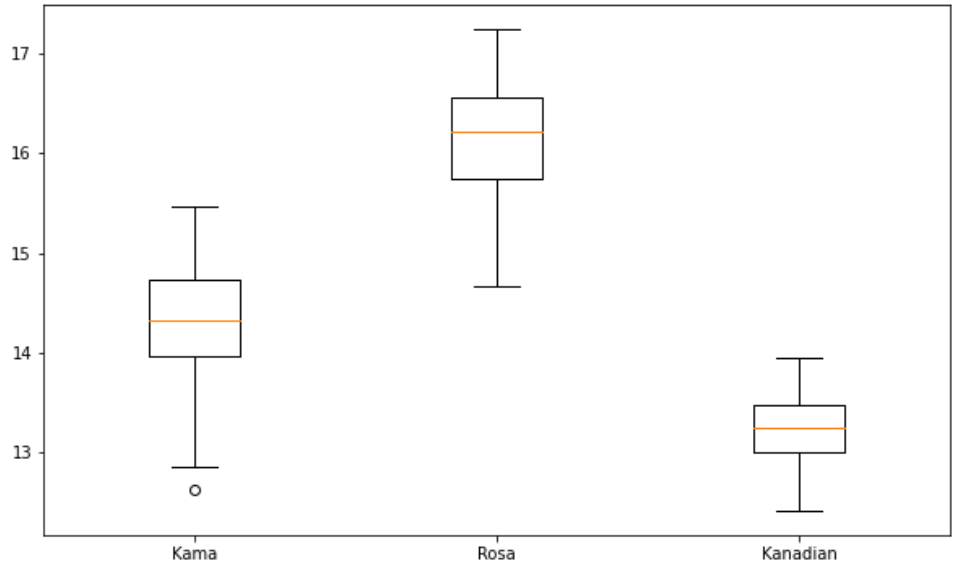

Figura 5: Boxplot della feature perimeter per varietà

1.3 Compactess $(C = 4 * \pi * A/P^2)$

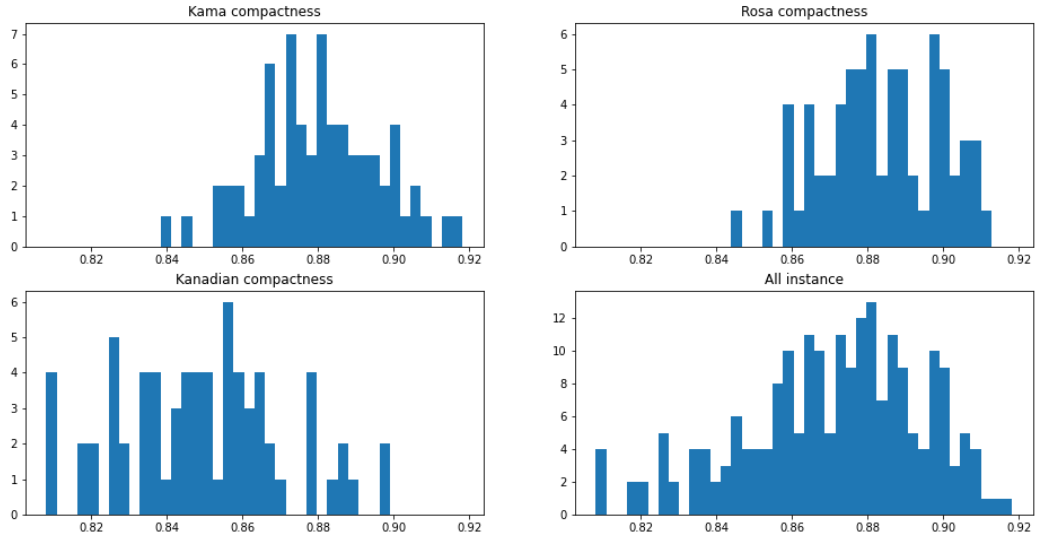

Figura 6: Istogramma della feature compactness per varietà e totale

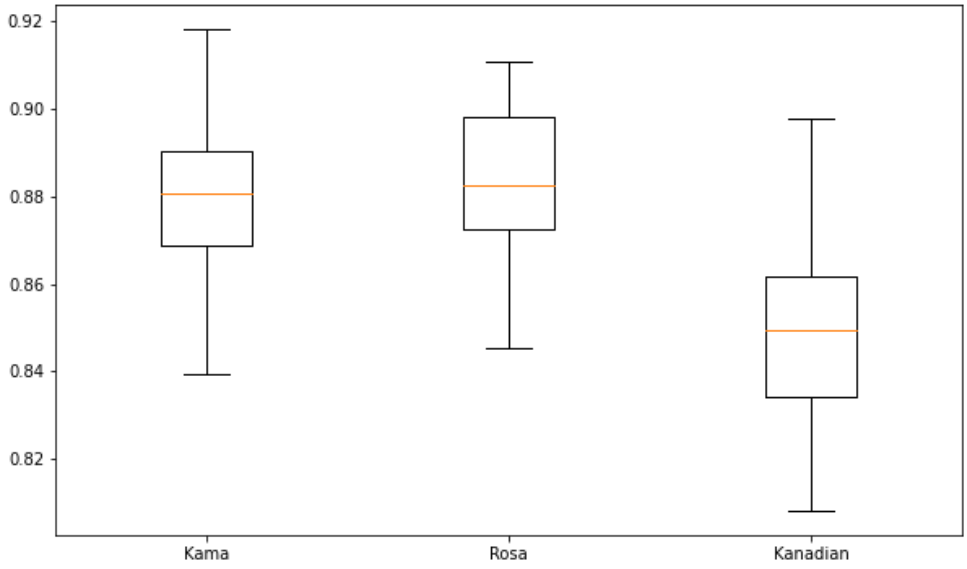

Figura 7: Boxplot della feature compactness per varietà

1.4 Length

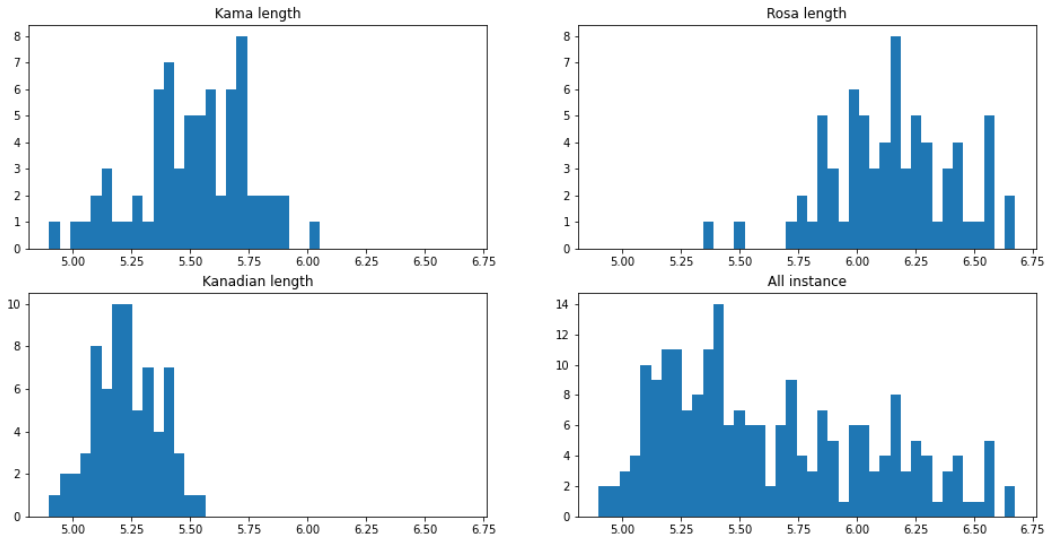

Figura 8: Istogramma della feature length per varietà e totale

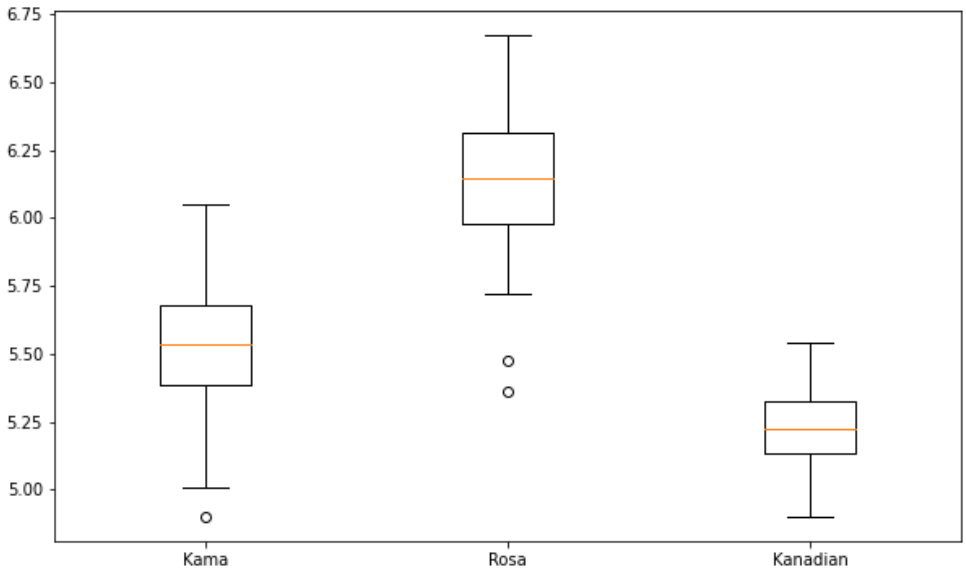

Figura 9: Boxplot della feature length per varietà

1.5 Width

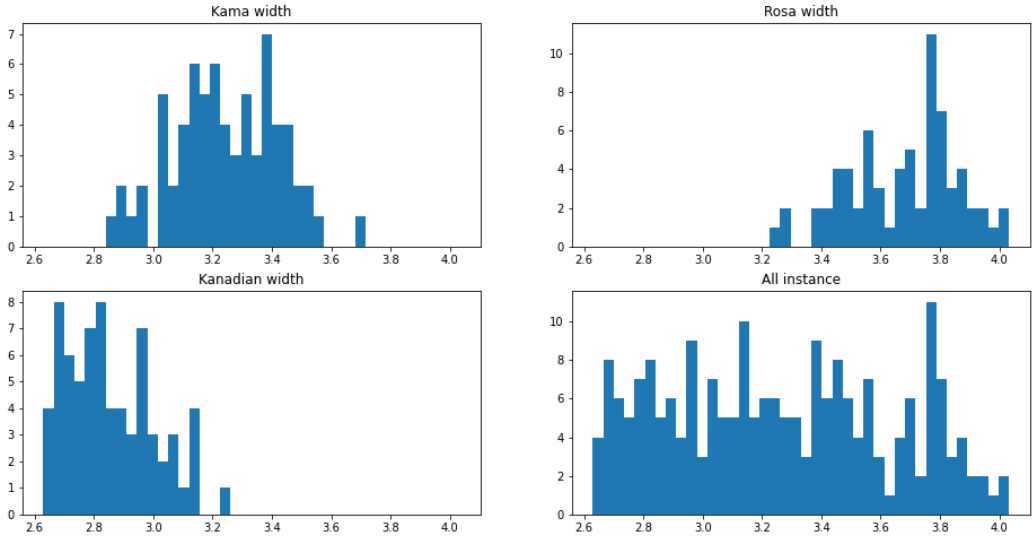

Figura 10: Istogramma della feature width per varietà e totale

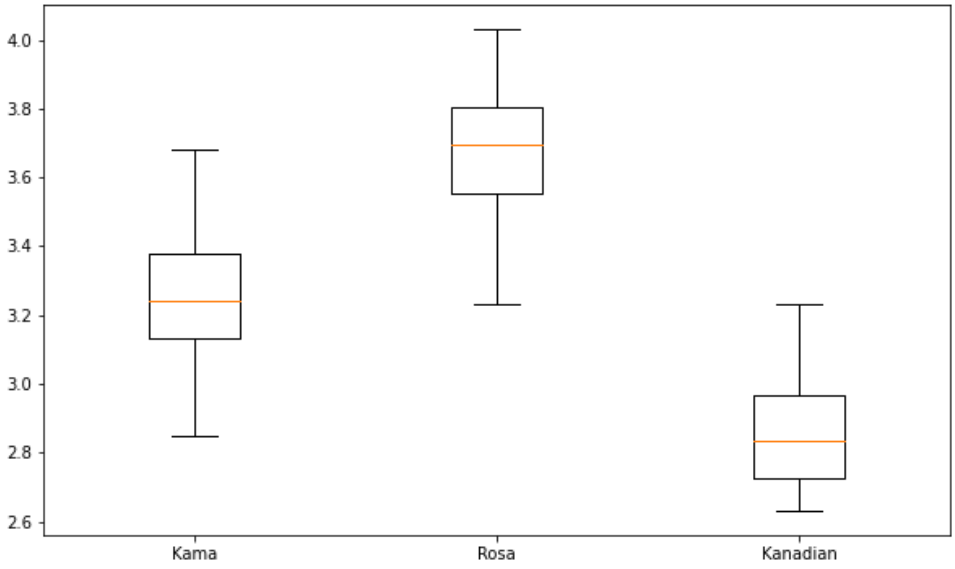

Figura 11: Boxplot della feature width per varietà

1.6 Asimmetry

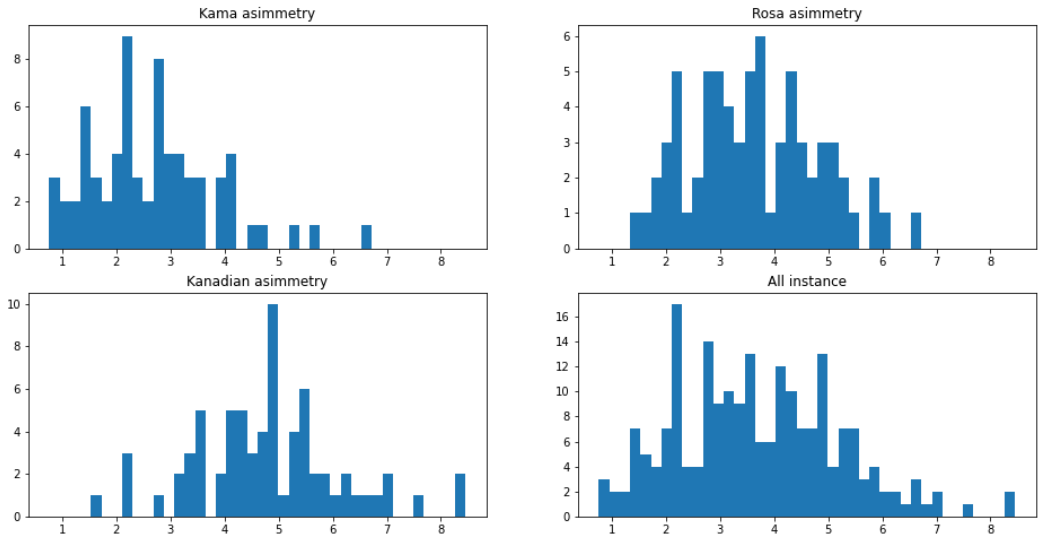

Figura 12: Istogramma della feature asimmetry per varietà e totale

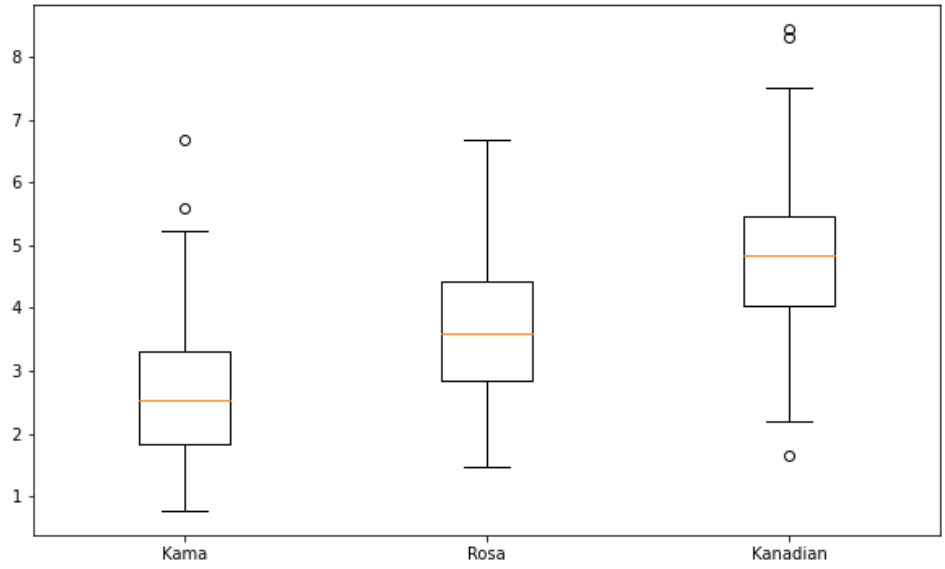

Figura 13: Boxplot della feature asimmetry per varietà

1.7 Groove

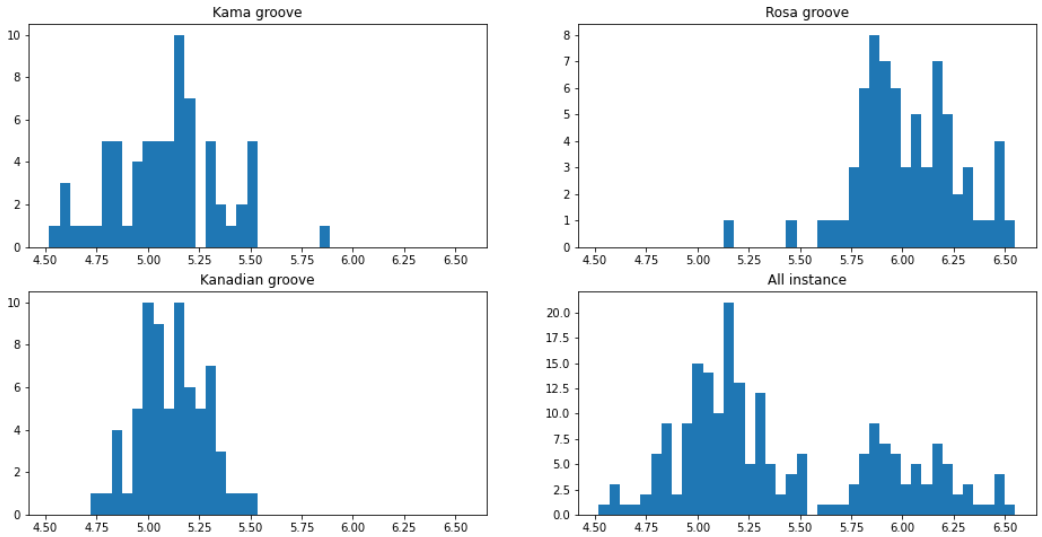

Figura 14: Istogramma della feature groove per varietà e totale

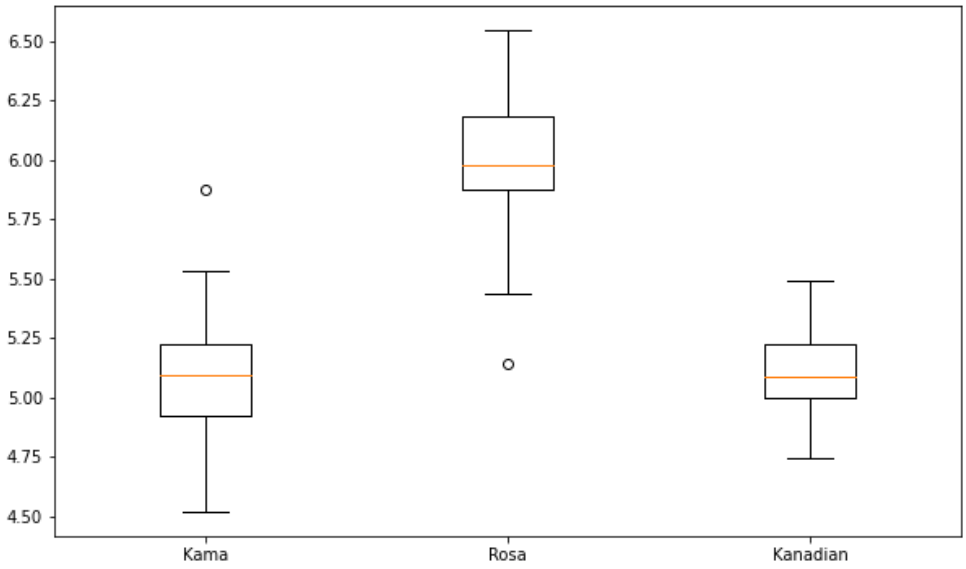

Figura 15: Boxplot della feature groove per varietà

2 Metodologia

Al fine di rendere i risultati replicabili è stato scelto un random state pari a 52.

Dopo aver verificato l'assenza di valori nulli o invalidi all'interno del dataset, esso è stato diviso in training set e test set con rapporto 80:20.

Sono stati utilizzati i seguenti classificatori:

- Alberi di decisione
- Random forest
- K Nearest Neighbors
- Adaboost

I classificatori sono stati addestrati e testati prima con i parametri di default e poi provando ad effettuare il tuning degli iperparamentri attraverso una grid search con valore di K per la cross-validation pari a tre, scelto basso per la ridotta dimensione del campione.

Per ciascun classificatore sono stati individuati i seguenti parametri relativi alle performance:

- Precisione
- Accuratezza
- Recall
- F1 score
- Matrice di confusione
- Importanza delle features

3 Alberi di decisione

Il classificatore basato su alberi di decisione genera un albero i cui nodi foglia rappresentano le classificazione mentre le ramificazione indicano i valori delle proprietà che portano alla predizione delle classificazioni. Il predicato contenuto in ciascun nodo interno indica il criterio di scelta ed è chiamato condizione di split, solitamente come parametro della condizione viene usato il misclassification error, l'indice di gini o la variazione di entropia.

3.1 Parametri di default

In tabella 1 vengono indicate le performance ottenute con i parametri di default e nella figura 16 la matrice di confusione sul test set.

Nella figura 17 viene indicata l'importanza data dal classificatore a ciascuna feature: l'importanza maggiore è data alle feature perimeter e groove, con le altre quasi irrilevanti e la feature area addirittura pari a zero.

Class	Precision	Recall	F-score	Support
Kama	0.66667	0.8	0.72727	10
Rosa	0.875	0.93333	0.90323	15
Kanadian	0.92857	0.76471	0.83871	17
macro avg	0.82341	0.83268	0.82307	42
avg	0.84708	0.83333	0.83522	42
accuracy	0.83333			

Tabella 1: Report classificazione per DecisionTreeClassifier default

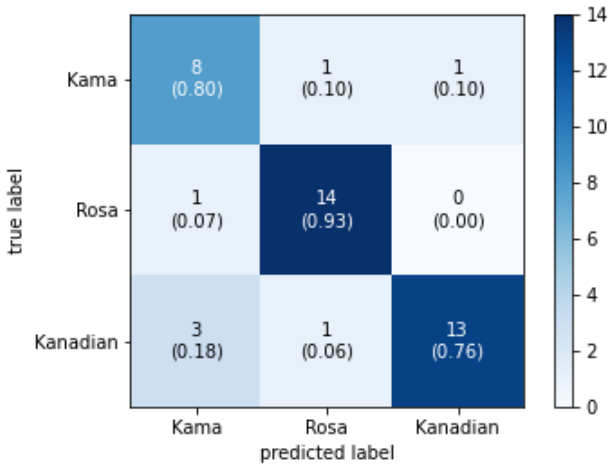

Figura 16: Matrice di confusione per DecisionTreeClassifier di default

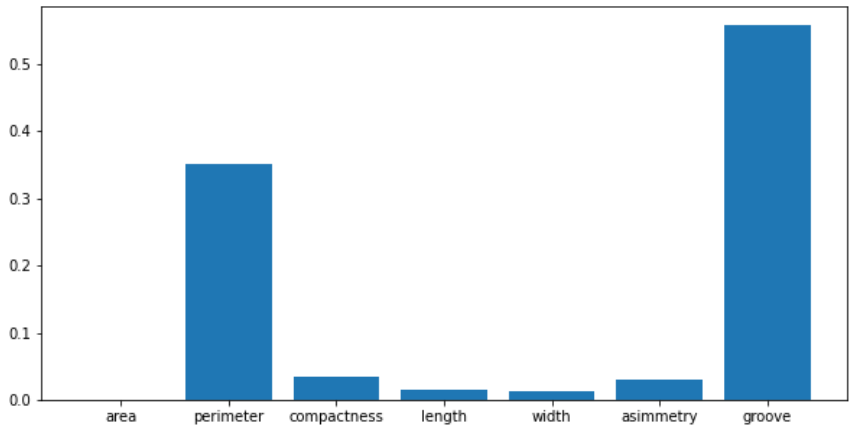

Figura 17: Importanza feature per DecisionTreeClassifier di default

3.2 Tuning degli iperparametri

I parametri cercati da gridsearch sono stati:

Criterion = gini Funzione che misura la qualità dello split

Splitter = random Strategia usata per scegliere lo split ad ogni nodo

Max features = auto Numero di feature che l'algoritmo considera nello scegliere il migliore split

Max depth = 2 Profondità massima dell'albero generato

L'accuratezza e la media degli altri parametri è migliorata, mentre l'importanza delle feature è cambiata molto avendo scelto come parametro di profondità massima 2: solo area e groove vengono prese in considerazione.

In tabella 2 vengono indicate le performance ottenute dopo il tuning dei parametri, nella figura 18 la matrice di confusione sul test set e nella figura 19 l'importanza data dal classificatore a ciascuna feature.

Class	Precision	Recall	F-score	Support
Kama	0.7	0.7	0.7	10
Rosa	0.93333	0.93333	0.93333	15
Kanadian	0.88235	0.88235	0.88235	17
macro avg	0.83856	0.83856	0.83856	42
avg	0.85714	0.85714	0.85714	42
accuracy	0.85714			

Tabella 2: Report classificazione per DecisionTreeClassifier tuned

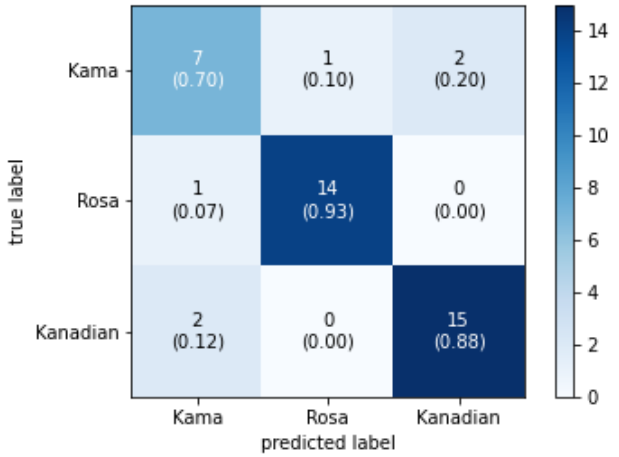

Figura 18: Matrice di confusione per DecisionTreeClassifier tuned

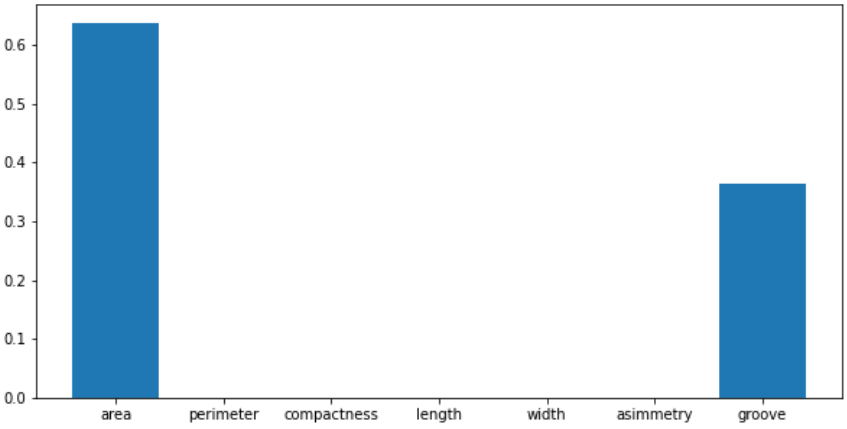

Figura 19: Importanza feature per DecisionTreeClassifier tuned

4 Random forest

Il classificatore basato su random forest è un metodo ensemble che consiste nell'aggregazione tramite bagging di alberi di decisione. Il bagging degli alberi consiste nel sampling ripetuto con sostituzione di elementi del training set per l'addestramento di ogni albero per poi fare la media delle previsioni di ciascun albero. Random forest aggiunge ulteriore randomizzazione a questo metodo andando a selezionare ad ogni split degli alberi un sottoinsieme casuale di feature per la decisione dello split.

4.1 Parametri default

In tabella 3 vengono indicate le performance ottenute con i parametri di default e nella figura 20 la matrice di confusione sul test set.

Nella figura 21 è visualizzata l'importanza data dal classificatore a ciascuna feature, possiamo subito notare che rispetto al metodo basato su alberi di decisione standard è data più importanza a ciascuna feature e nessuna viene totalmente ignorata.

Class	Precision	Recall	F-score	Support
Kama	0.72727	0.8	0.7619	10
Rosa	0.93333	0.93333	0.93333	15
Kanadian	0.9375	0.88235	0.90909	17
macro avg	0.86604	0.8719	0.86811	42
avg	0.88596	0.88095	0.8827	42
accuracy	0.88095			

Tabella 3: Report classificazione per RandomForestClassifier default

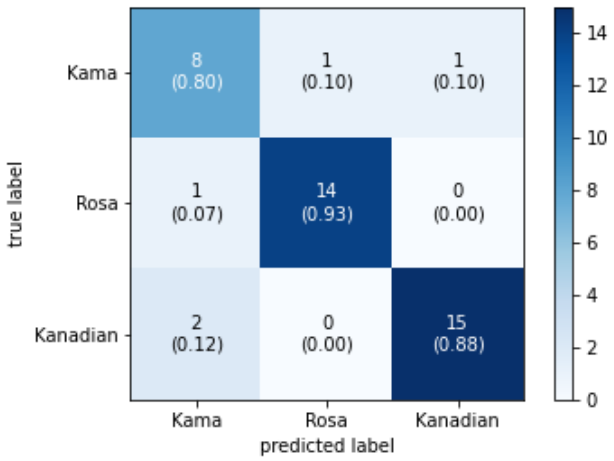

Figura 20: Matrice di confusione per RandomForestClassifier di default

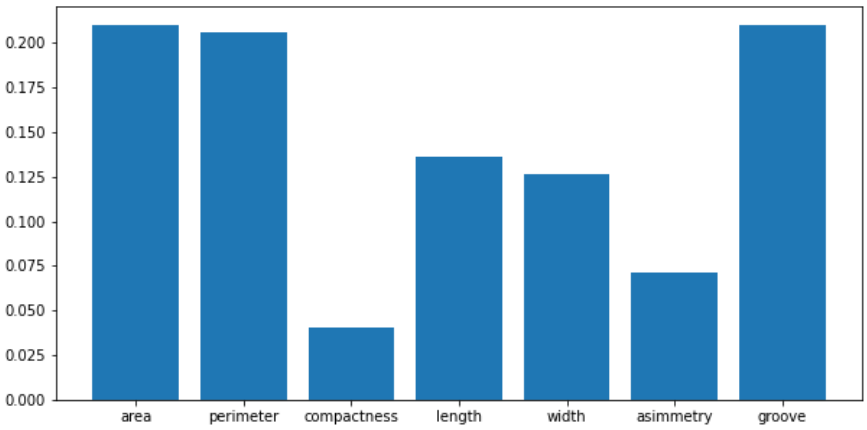

Figura 21: Importanza feature per RandomForestClassifier di default

4.2 Tuning dei parametri

I parametri cercati da gridsearch sono stati:

Criterion = entropy Funzione che misura la qualità dello split

Bootstrap = false Indica se gli esemplari già considerati dall'algoritmo possono riapparire nelle future iterazioni

Class weight = balanced Indica il peso associato a ciascuna classe, balanced significa che viene aggiustato proporzionalmente alla frequenza della classe nell'input

Max depth = 7 Profondità massima degli alberi generati

In tabella 4 vengono indicate le performance ottenute con gli iperparametri migliori trovati, nella figura 22 la matrice di confusione sul test set e nella figura 23 l'importanza data dal classificatore a ciascuna feature.

L'accuratezza e gli altri parametri di performance sono aumentati, mentre si può vedere che l'importanza delle feature è stata distribuita più su alcune che altre, senza però ignorarne completamente o quasi nessuna.

Class	Precision	Recall	F-score	Support
Kama	0.8	0.8	0.8	10
Rosa	0.93333	0.93333	0.93333	15
Kanadian	0.94118	0.94118	0.94118	17
macro avg	0.8915	0.8915	0.8915	42
avg	0.90476	0.90476	0.90476	42
accuracy	0.90476			

Tabella 4: Report classificazione per RandomForestClassifier tuned

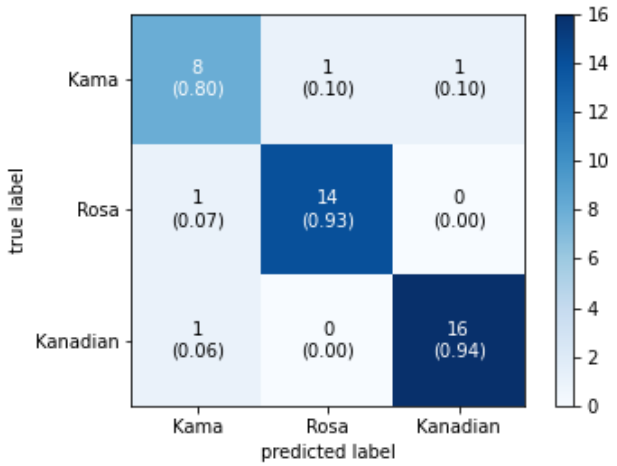

Figura 22: Matrice di confusione per RandomForestClassifier tuned

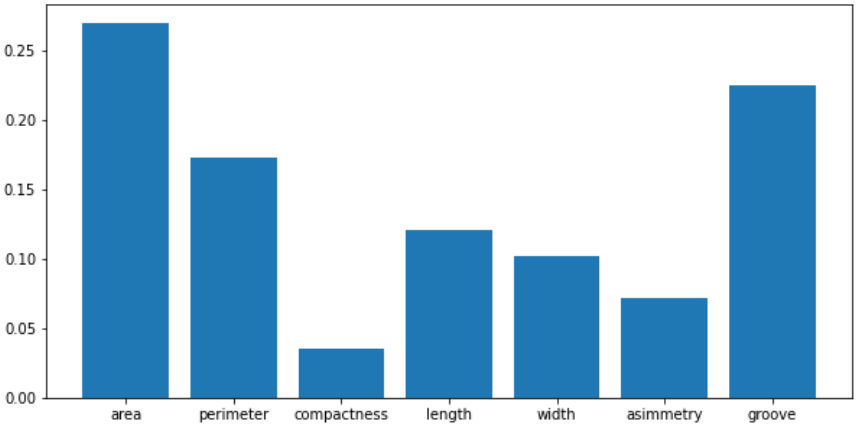

Figura 23: Importanza feature per RandomForestClassifier tuned

5 K Nearest Neighbors

L'algoritmo di classificazione KNN consiste nell'assegnare una classe ad un oggetto guardando la maggioranza delle classi dei k soggetti utilizzati per il training che risultano più vicini ad esso nello spazio delle features. Questo metodo non prevede la definizione di un modello di classificazione ma utilizza direttamente la somiglianza con le istanze di training per classificarne di nuove, risulta quindi abbastanza efficiente computazionalmente con lo svantaggio di poter considerare solamente features di tipo metrico.

5.1 Parametri di default

In tabella 5 vengono indicate le performance ottenute con i parametri di default, nella figura 24 la matrice di confusione sul test set. L'importanza delle feature per questo algoritmo è sempre uguale per tutte perchè valuta la distanza tra i punti considerandole tutte.

Già con i parametri di default risulta il miglior classificatore considerato.

Class	Precision	Recall	F-score	Support
Kama	0.88889	0.8	0.84211	10
Rosa	0.93333	0.93333	0.93333	15
Kanadian	0.94444	1.0	0.97143	17
macro avg	0.92222	0.91111	0.91562	42
avg	0.92725	0.92857	0.92703	42
accuracy	0.92857			

Tabella 5: Report classificazione per KNeighborsClassifier default

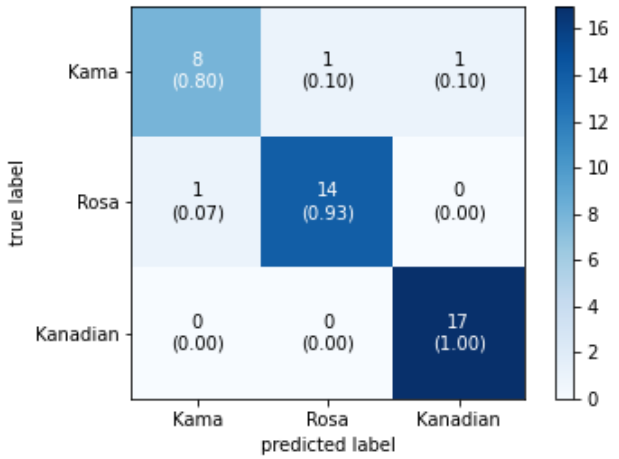

Figura 24: Matrice di confusione per KNeighborsClassifier di default

5.2 Tuning dei parametri

I parametri cercati da gridsearch sono stati:

Neighbors = 5 Numero di vicini considerati nella classificazione

Weights = distance Peso utilizzato nella predizione, distance indica che il peso è l'inverso della distanza tra i punti

Algorithm = ball tree Algoritmo usato per trovare i punti vicini

P = 4 Potenza usata nella metrica di Minkowski per stabilire la distanza tra soggetti

In tabella 6 vengono indicate le performance ottenute con gli iperparametri migliori trovati, nella figura 25 la matrice di confusione sul test set.

L'accuratezza e gli altri parametri di performance sono aumentati, raggiungendo il massimo tra i classificatori considerati.

Class	Precision	Recall	F-score	Support
Kama	1.0	0.8	0.88889	10
Rosa	0.9375	1.0	0.96774	15
Kanadian	0.94444	1.0	0.97143	17
macro avg	0.96065	0.93333	0.94269	42
avg	0.95519	0.95238	0.95046	42
accuracy	0.95238			

Tabella 6: Report classificazione per KNeighborsClassifier tuned

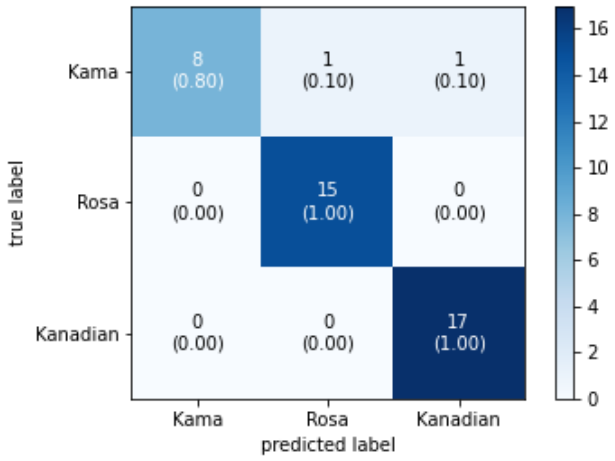

Figura 25: Matrice di confusione per KNeighborsClassifier tuned

6 AdaBoost

Il classificatore AdaBoost, o adaptive boosting, è un meta algoritmo ensemble che consiste nell'utilizzare l'output di altri classificatori combinandoli in una media pesata dalla difficoltà di classificazione delle singole istanze. Ad ogni iterazione i nuovi classificatori deboli sono ottimizzati per riconoscere le istanze più difficili e quindi sopperiscono alle mancanze dei precedenti.

6.1 Parametri standard

In tabella 7 vengono indicate le performance ottenute con i parametri di default, nella figura 26 la matrice di confusione sul test set e nella figura 27 l'importanza data dal classificatore a ciascuna feature.

L'accuratezza non è molto elevata rispetto agli altri, inoltre l'algoritmo considera praticamente soltanto le features perimeter e compactness, ignorando o quasi le altre.

Class	Precision	Recall	F-score	Support
Kama	0.6	0.6	0.6	10
Rosa	0.92857	0.86667	0.89655	15
Kanadian	0.83333	0.88235	0.85714	17
macro avg	0.7873	0.78301	0.78456	42
avg	0.81179	0.80952	0.80999	42
accuracy	0.80952			

Tabella 7: Report classificazione per AdaBoost default

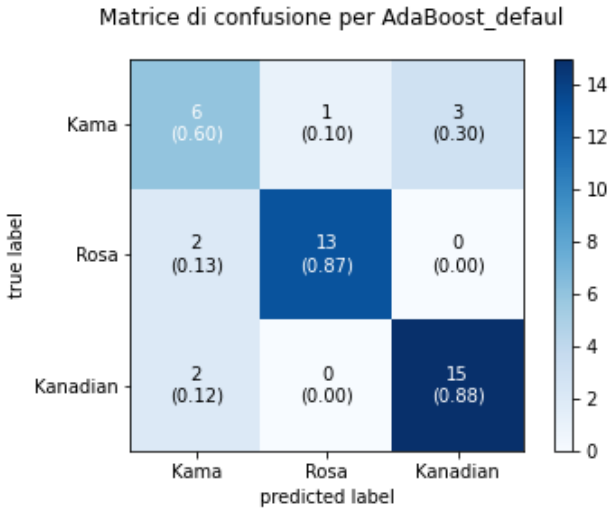

Figura 26: Matrice di confusione per AdaBoostClassifier di default

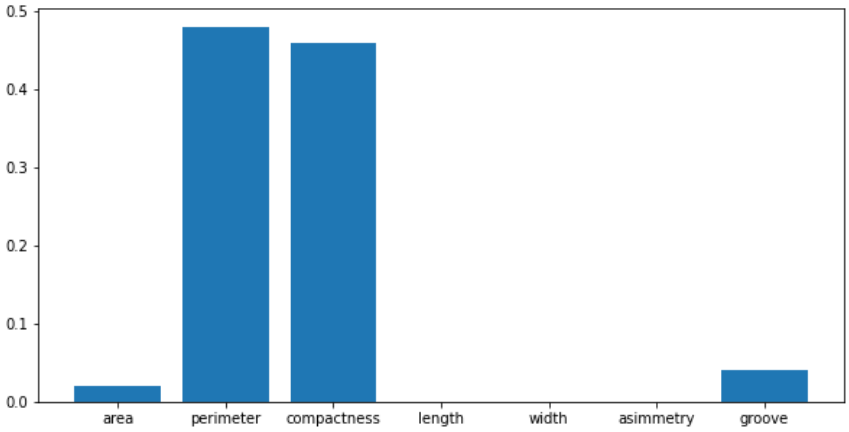

Figura 27: Importanza feature per AdaBoostClassifier di default

6.2 Tuning dei parametri

I parametri cercati da gridsearch sono stati:

N estimators = 40 Numero massimo di estimatori impiegati nella procedura di boosting

Learning rate = 0.5 Peso associato ad ogni classificatore per ogni iterazione di boosting

In tabella 8 vengono indicate le performance ottenute con gli iperparametri migliori trovati, nella figura 28 la matrice di confusione sul test set e nella figura 29 l'importanza data dal classificatore a ciascuna feature.

L'accuratezza e i parametri di performance sono aumentati di molto malgrado la modifica leggera di due soli iperparametri, inoltre l'importanza delle features è più bilanciata e l'unica feature esclusa è la lunghezza.

Class	Precision	Recall	F-score	Support
Kama	0.69231	0.9	0.78261	10
Rosa	0.93333	0.93333	0.93333	15
Kanadian	1.0	0.82353	0.90323	17
macro avg	0.87521	0.88562	0.87306	42
avg	0.90293	0.88095	0.88526	42
accuracy	0.88095			

Tabella 8: Report classificazione per AdaBoost tuned

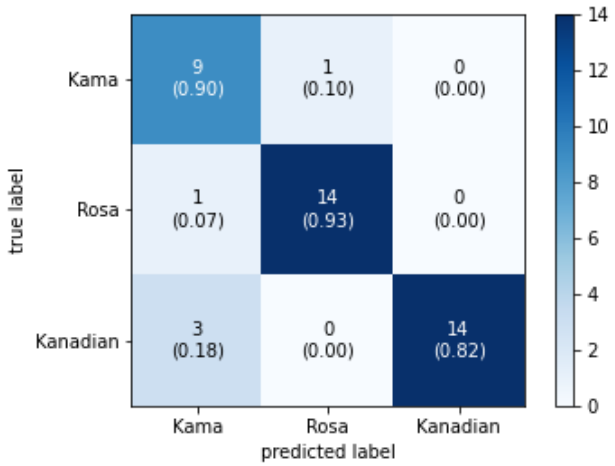

Figura 28: Matrice di confusione per AdaBoostClassifier tuned

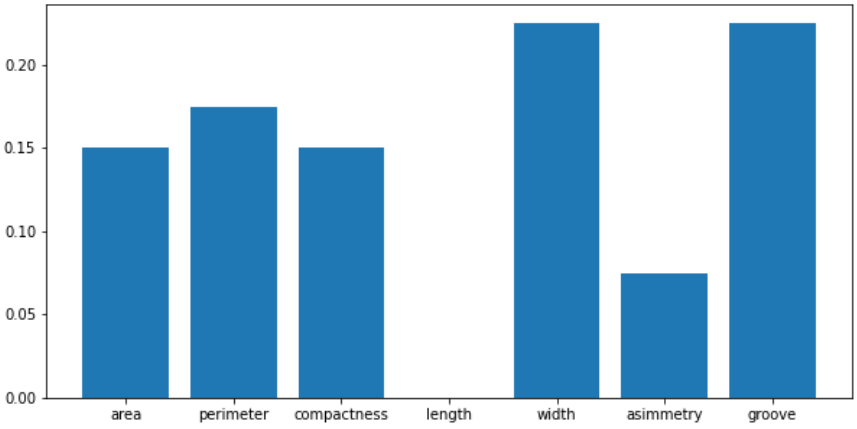

Figura 29: Importanza feature per AdaBoostClassifier tuned

Tabella 9: Report classificazione per Neural networks classifier

Class	Precision	Recall	F-score	Support
Kama	0.71429	1.0	0.83333	10
Rosa	1.0	1.0	1.0	15
Kanadian	1.0	0.76471	0.86667	17
macro avg	0.90476	0.92157	0.9	42
avg	0.93197	0.90476	0.90635	42
accuracy	0.90476			

7 Reti neurali

La classificazione basata su reti neurali crea una rete di nodi con associato a ciascuno una funzione di attivazione secondo una determinata soglia e determinati pesi per fare passare l'informazione tra un nodo e il livello successivo.

Dopo qualche esperimento manuale ho scelto di utilizzare una rete con un solo livello nascosto, data la ridotta dimensione del dataset, con funzione di attivazione relu, mentre per il livello di output funzione di attivazione softmax. Ho utilizzato la loss function sparse categorical crossentropy e l'ottimizzatore adam che ha dato risultati migliori del sdg.

In tabella 9 sono riportate le perfomance del classificatore che raggiunge un accuratezza di 0.90476, valore buono ma comunque inferiore al knn, probabilmente perchè la ridotta dimensione del dataset non permette di sfruttare appieno la complessità del classificatore. La figura 30 mostra invece la matrice di confusione.

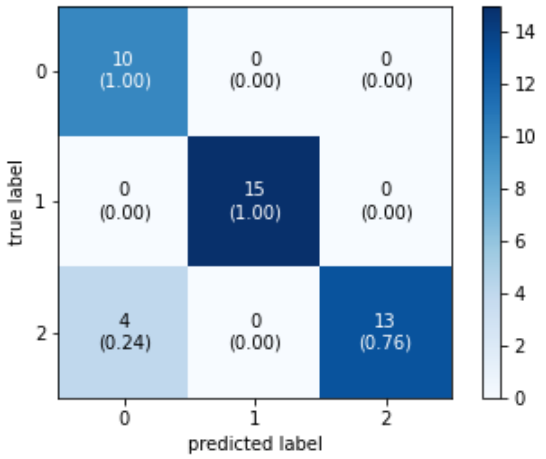

Figura 30: Matrice di confusione per il classificatore basato su neural networks

8 Autosklearn

Ho provato infine ad utilizzare il tool di classificazione automatica autosklearn per vedere quali altri classificatori non esaminati potessero dare risultati migliori, con un tempo limite per l'algoritmo di 20 minuti. Il migliore è risultato il multi layer perceptron con un accuratezza massima di 1, bisognerebbe quindi prenderlo in esame singolamente per verificare la presenza di overfitting e andare a studiarlo ottimizzandone i parametri con gridsearch e una cross-fold validation.

In figura 31 vengono visualizzati i primi 50 classificatori trovati

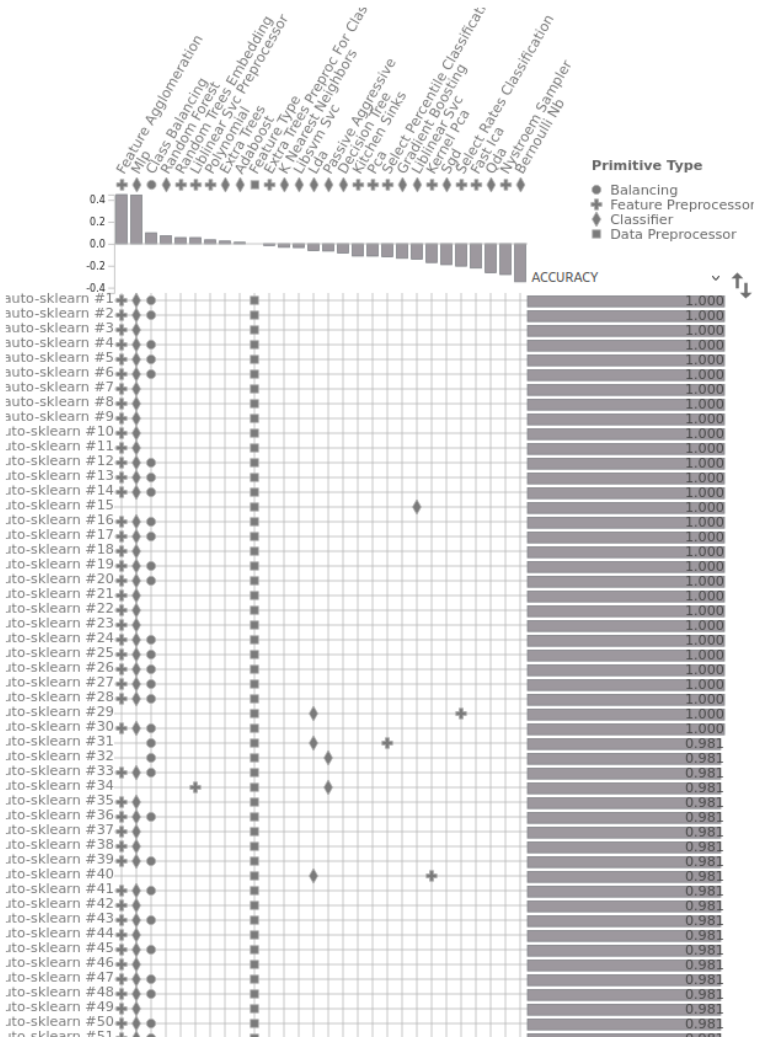

Figura 31: Risultato del tool autosklearn

9 Conclusioni

In tabella 10 vengono riportati gli indicatori di performance di tutti i classificatori utilizzati, dopo il tuning degli iperparametri: tutti hanno raggiunto un'accuratezza di almeno 85%.

Il migliore risulta il classificatore KNN mentre il peggiore il classificatore basato su alberi di decisione.

Come prevedibile le random forest si comportano meglio degli alberi di decisione essendo un'affinamento di questi ultimi, inoltre come si evince dalla figura 32 che riassume l'importanza delle features per i classificatori random forest essendo più robusto all'overfitting può considerare più feature mentre per gli alberi di decisione nel processo di tuning degli iperparametri si è deciso di arrivare solo a profondità due e quindi considerare solo due feature, probabilmente per l'apparizione di overfitting durante la validazione.

Per quanto riguarda AdaBoost, il risultato lascia a desiderare considerando anche la relativa complessità del classificatore rispetto agli altri: in questo caso probabilmente la dimensione ridotta del dataset ha sfavorito la precisione del modello.

Lo stesso discorso di AdaBoost potrebbe essere applicato al classificatore basato su rete neurale, anche questo non riesce ad esprimere il suo potenziale dato dalla sua complessità a causa del dataset ridotto.

Il classificatore k-nearest-neighbors risulta il migliore con 95% di accuratezza: in suo favore ha giocato la buona differenza di distribuzione dei valori delle features tra le tre classi e il piccolo numero di outliers presenti che si può osservare nei grafici della sezione 1, noltre il ridotto numero di istanze del dataset di training non influisce la precisione di questo classificatore, essendo che considera sempre solo k vicini per la classificazione di nuovi oggetti.

Classificatore	Precisione	Recupero	F score	Accuratezza
Decision Tree	0.85714	0.85714	0.85714	0.85714
Random Forest	0.90476	0.90476	0.90476	0.90476
KNN	0.95519	0.95238	0.95046	0.95238
AdaBoost	0.90293	0.88095	0.88526	0.88095
Neural Network	0.93197	0.90476	0.90635	0.90476

Tabella 10: Tabella riassuntiva degli indicatori di performance dei classificatori

Figura 32: Confronto tra le importanze delle features

A Usage Report

Colab notebook dell'elaborato:
https://colab.research.google.com/drive/1ahxCzHPt6fNJw7GKqna77A12KcNt0d1h?authuser=2#scrollTo=RT2LHxzxcSCk

L'elaborato è stato prodotto usando i seguenti software e librerie:
Google Colab
https://colab.research.google.com/
Python
https://www.python.org/
Sklearn
https://scikit-learn.org/stable/
Pandas
https://pandas.pydata.org/
Numpy
https://numpy.org/
Seaborn
https://seaborn.pydata.org/ Keras
https://keras.io/ Autosklearn
https://automl.github.io/auto-sklearn/master/
I dati sono stati presi dal database UCI, forniti dall'istituto di agrofisica dell'università di Lublino:
UCI
https://archive.ics.uci.edu/ml/index.php
Dataset 'seeds'
https://archive.ics.uci.edu/ml/datasets/seeds
Institute of Agrophysics of the Polish Academy of Sciences in Lublin
https://www.ipan.lublin.pl/en/

CopyRight