Predizione classi IRIS tramite Modelli di Classificazione e Apprendimento

Documentazione Progetto

Gruppo di lavoro

- Andrea Mastrolonardo, 716698, a.mastrolonardo5@studenti.uniba.it
- Enrico Pappalardo, 727662, e.pappalardo1@studenti.uniba.it

https://github.com/EnricoP-02/Progetto-ICon

AA 2022-2023

Introduzione

Il nostro dominio di interesse è rappresentato dal dataset IRIS, il quale è stato creato da R.A. Fisher ed è spesso utilizzato per l'applicazione di metodi di classificazione tramite modelli di apprendimento supervisionato e non. Il dataset contiene un attributo su cui eseguire la predizione, ovvero la tipologia di pianta iris (3 classi con 50 istanze ciascuna) e 4 attributi in ogni istanza di fiore su cui addestrare i modelli scelti per la classificazione.

Gli attributi presenti nel dataset sono i seguenti:

- Sepal length [lunghezza del sepalo] (in cm)
- Sepal width [larghezza del sepalo] (in cm)
- Petal length [lunghezza del petalo] (in cm)
- Petal width [larghezza del petalo] (in cm)
- Classe (iris setosa, iris versicolour, iris virginica)

Sommario

Il Knowledge Base System (KBS) da noi utilizzato è composto principalmente da due aspetti: la Knowledge Base (KB) e la classificazione delle classi IRIS.
Per lo scheletro della KB abbiamo deciso di utilizzare l'implementazione scritta interamente in python eseguita dall'utente XinTongBUPT: https://github.com/XinTongBUPT/Knowledge-Base.

La KB prende in input un file di testo contenente alcune regole e affermazioni da noi scritte (basandoci sui valori da alcune istanze del dataset IRIS), per poi inferire sulle affermazioni, ottenendo le informazioni chieste durante la fase di test. La KB possiede 3 principali azioni: Assert, Ask e Retract. Assert serve per aggiungere affermazioni e regole alla KB, Ask serve per chiedere query e ottenere affermazioni durante la fase di test e Retract serve per rimuovere affermazioni o regole dalla KB.

La classificazione delle classi è eseguita con più modelli supervisionati e non, elencati nel prossimo paragrafo. Una volta eseguita la classificazione, i risultati delle metriche per ogni classe vengono aggiunti come affermazioni all'interno di un file, poi letto dalla KB, la quale aggiunge queste come sua conoscenza. Successivamente verranno eseguiti test su queste affermazioni aggiunte per verificare che la KB abbia assimilato la conoscenza correttamente.

Il processo di classificazione viene eseguito dalla KB sul dataset IRIS e su tre versioni modificate del dataset da noi costruite (Iris_Setosa, Iris_Versicolour e Iris_Virginica). Per ciascun dataset modificato gli elementi al suo interno appartengono a due classi; ad esempio, per il dataset Iris_Setosa, le classi al suo interno sono: Iris-setosa, Iris-not-setosa.
Queste tre versioni modificate servono per analizzare il comportamento dei vari modelli da noi implementati nel caso in cui la classificazione è binaria (esempio: un elemento i è di classe A oppure B) e confrontarlo nel caso in cui la classificazione è sulla tre classi origninali del dataset. La classificazione avviene tramite il comando Ask inserendo il termine classification e il nome di una delle classi (setosa, virginica e versicolour) se si vuole calcolare la classificazione sul dataset modificato (contenente solo due classi) corrispondente alla classe inserita o il termine all per calcolare la classificazione su tutte e tre le classi contenute nel dataset.

Elenco argomenti di interesse

- Case-Based Reasoning: KNN (Apprendimento Supervisionato)
- Decision-Tree (Apprendimento Supervisionato)
- Random Forest (Apprendimento Supervisionato)
- Logistic Regressor (Apprendimento Supervisionato)
- Neural Network (Apprendimento Supervisionato)
- Naive Bayes (Apprendimento Probabilistico)
- Hard Clustering: K-Means (Apprendimento Non Supervisionato)

K-Nearest Neighbors

Sommario

La classificazione si esegue dividendo gli attributi all'interno del dataset in due gruppi: gli attributi di input di cui si conoscono sempre i valori e gli attributi di output su cui effettuare la classificazione.

Nel nostro caso gli attributi di input sono i seguenti: sepal length (lunghezza del sepalo), sepal width (larghezza del sepalo), petal length (lunghezza del petalo) e pedal width (larghezza del petalo).

L'attributo di output invece, nel nostro caso sarà la classe del fiore (setosa, versicolour e virginica nel caso di classificazione multi-classe e, ad esempio, setosa/non setosa nel caso di classificazione binaria).

Successivamente, il dataset viene diviso, tramite la procedura train_test_split in 4 set di istanze: X_train, X_test, y_train, y_test.

Una volta eseguito il processo di classificazione, si crea un file classification_statements, il quale contiene le asserzioni derivate dai risultati della classificazione, successivamente la KB legge questo file e ottiene, tramite l'azione Assert, le metriche relative alla classificazione effettuata dal modello K-Nearest Neighbors (KNN) associate alla classe o classi su cui si è effettuata.

Strumenti utilizzati

Il classificatore K-Nearest Neighbors è un caso di Case-Based Reasoning nel quale, dato un esempio, si cercano i k esempi più simili memorizzati al fine di predire la classe, cioè la feature target nel nostro caso. La classe verrà assegnata osservando qual'è la classe a cui appartiene la maggioranza degli esempi più simili al nostro esempio da predire.

La procedura K-Fold Cross Validation divide tutti gli esempi considerati in k gruppi (dette folds) di una stessa dimensione se possibile, utilizzando k-1 folds per l'addestramento del classificatore e la rimanente per testarlo.

Decisioni di Progetto

Per implementare il modello K-Nearest Neighbors abbiamo utilizzato la libreria sklearn.neighbors.KNeighborsClassifier e, per migliorare la sua efficacia abbiamo implementato la procedura K-Fold CrossValidation tramite la funzione GridSearchCV della libreria sklearn.model_selection.

K-Nearest Neighbors:
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

GridSearchCV:
https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Valutazione

Le metriche utilizzate per misurare l'efficacia della classificazione eseguita dal KNN sono le seguenti: Accuracy, F1 Score, Precision, Recall, Balanced Accuracy e l'AUC Score.

Tutte le metriche utilizzate fanno parte del modulo sklearn.metrics: https://scikit-learn.org/stable/modules/model_evaluation.html

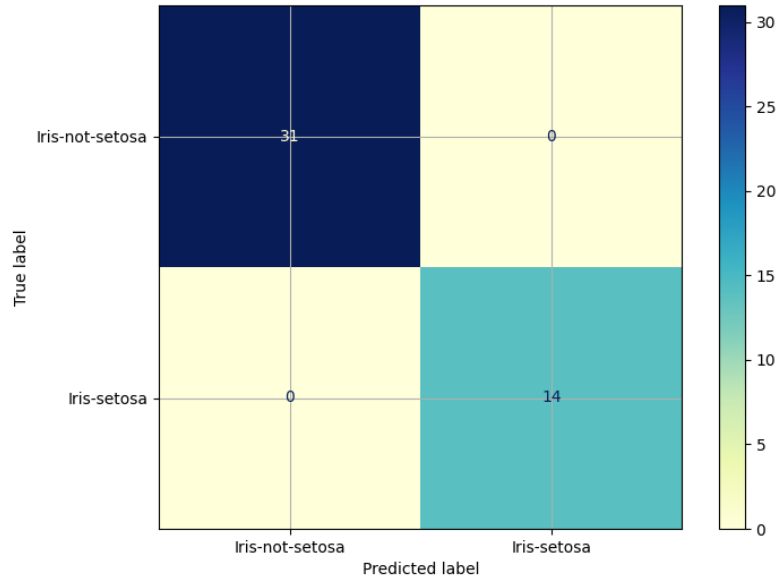
Aggiunti i valori delle metriche nella KB, esse posso essere oggetto di query eseguite dall'azione Ask.

Per ogni modello abbiamo costruito una matrice di confusione al fine di rappresentare l'accuratezza della classificazione. Le colonne della matrice rappresentano i valori predetti e le righe rappresentano i valori reali ottenuti dal processo di classificazione. L'elemento, ad esempio, sulla riga i e colonna j è il numero di casi in cui il modello ha classificato la classe i come classe j.

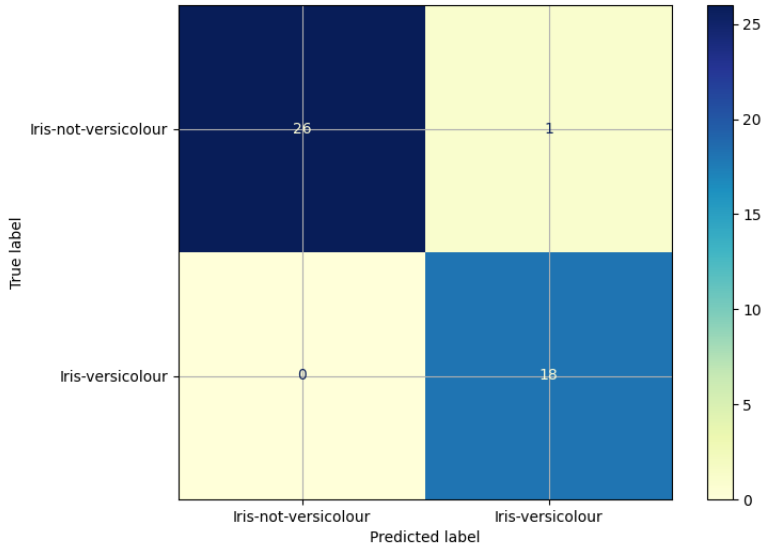
I valori mostrati nelle successive tabelle e grafici potrebbero variare leggermente.

La classificazione avviene su un test set costituito da 45 esempi (il 30% del dataset).

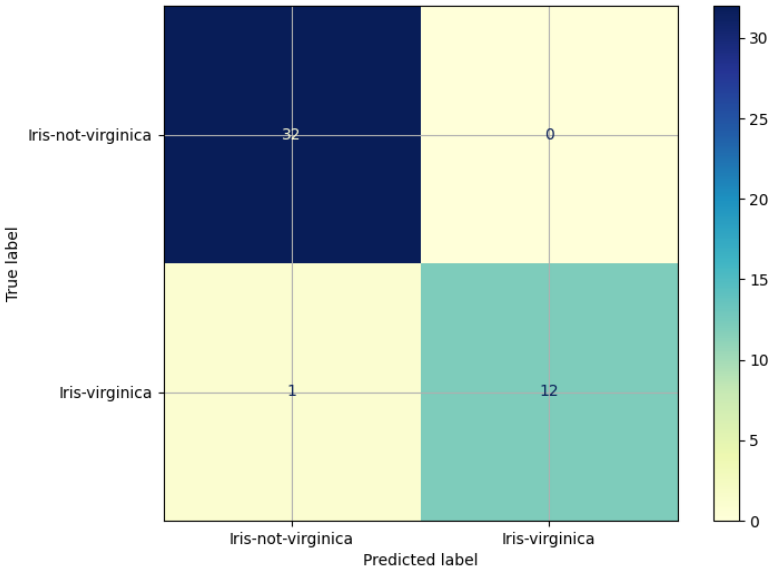
Per la classificazione eseguita sul dataset Iris_Setosa, Iris_Versicolour e Iris_Virginica i valori delle metriche prodotti dalla classificazione del modello K-Nearest Neighbors sono i seguenti:



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
K Nearest Neighbors	1.0	1.0	1.0	1.0	1.0	1.0



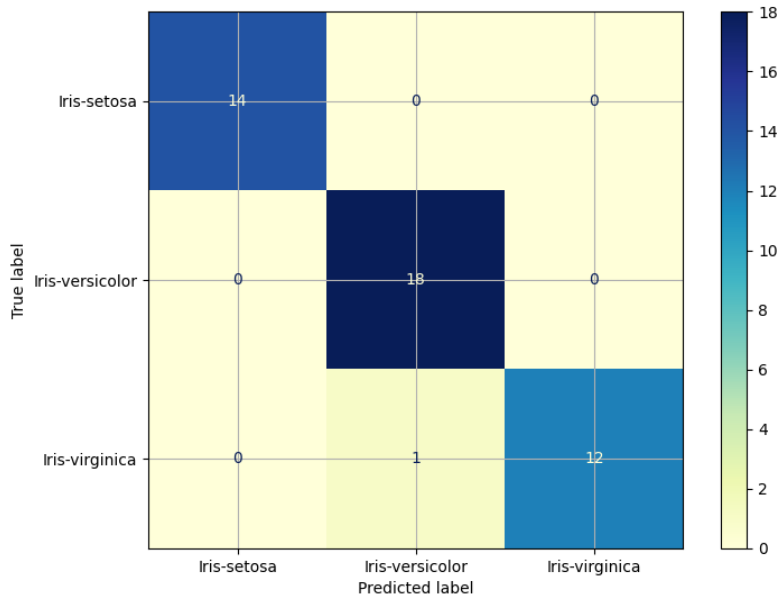
	accuracy	f1_score	precision	recall	balanced_accuracy	auc
K Nearest Neighbors	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9814814814814814	0.9958847736625515



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
K Nearest Neighbors	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9615384615384616	0.9987980769230769

Nel caso di Iris_Setosa, il KNN classifica correttamente le classi di tutti gli esempi; negli altri due casi, invece, non riesce ad ottenere un'accuratezza perfetta.
Questo risultato potrebbe variare con più esecuzioni della classificazione.

Per quanto riguarda la classificazione del dataset originale Iris, i risultati sono i seguenti:



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
K Nearest Neighbors	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9743589743589745	0.9970256937849532

Anche in questo caso la KNN non riesce ad ottenere un'accuratezza perfetta, ma il valore di ciascuna metrica (anche se diminuito circa del 3% rispetto ai valori osservati nei risultati della classificazione binaria su Iris_Setosa) ci conferma che l'errore è minimale e la classificazione effettuata da questo modello, in tutti i casi, è ottima.

Decision Tree

Sommario

La classificazione si esegue dividendo gli attributi all'interno del dataset in due gruppi: gli attributi di input di cui si conoscono sempre i valori e gli attributi di output su cui effettuare la classificazione.

Nel nostro caso gli attributi di input sono i seguenti: sepal length (lunghezza del sepalo), sepal width (larghezza del sepalo), petal length (lunghezza del petalo) e pedal width (larghezza del petalo).

L'attributo di output invece, nel nostro caso sarà la classe del fiore (setosa, versicolour e virginica nel caso di classificazione multi-classe e, ad esempio, setosa/non setosa nel caso di classificazione binaria).

Successivamente, il dataset viene diviso, tramite la procedura train_test_split in 4 set di istanze: X_train, X_test, y_train, y_test.

Una volta eseguito il processo di classificazione, si crea un file classification_statements, il quale contiene le asserzioni derivate dai risultati della classificazione, successivamente la KB legge questo file e ottiene, tramite l'azione Assert, le metriche relative alla classificazione effettuata dal modello Decision Tree associate alla classe o classi su cui si è effettuata.

Strumenti utilizzati

Dato un Decision Tree, cioè albero di decisione (costituito da nodi interni contenenti istuzioni condizionali if,then,else e nodi foglia contenenti la predizione della classe) e esempi contenenti le feature di input, la classificazione avviene partendo dalla radice percorrendo gli archi corrispondenti al risultato della condizione valutata all'interno del nodo intermedio incontrato fino a raggiungere un nodo foglia, il quale conterrà la predizione (la classe dell'istanza esaminata).

La procedura K-Fold Cross Validation, come già riportato, divide tutti gli esempi considerati in k gruppi (dette folds) di una stessa dimensione se possibile, utilizzando k-1 folds per l'addestramento del classificatore e la rimanente per testarlo.

Decisioni di Progetto

Per implementare il modello degli alberi di decisione abbiamo utilizzato il modulo DecisionTreeClassifier dalla libreria sklearn.tree e, per migliorare la sua efficacia abbiamo implementato la procedura K-Fold CrossValidation tramite la funzione GridSearchCV della libreria sklearn.model_selection.

DecisionTreeClassifier:
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

GridSearchCV:
https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Valutazione

Le metriche utilizzate per misurare l'efficacia della classificazione eseguita dall'albero di decisione sono le stesse usate nella valutazione del KNN, ovvero le seguenti: Accuracy, F1 Score, Precision, Recall, Balanced Accuracy e l'AUC Score.

Tutte le metriche utilizzate fanno parte del modulo sklearn.metrics: https://scikit-learn.org/stable/modules/model_evaluation.html

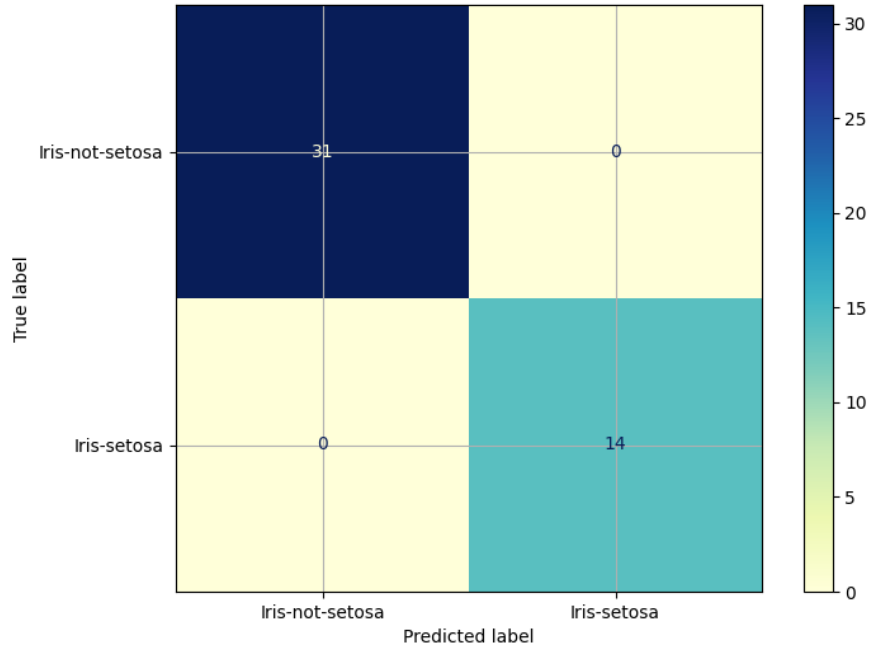
Aggiunti i valori delle metriche nella KB, esse posso essere oggetto di query eseguite dall'azione Ask.

Per ogni modello abbiamo costruito una matrice di confusione al fine di rappresentare l'accuratezza della classificazione. Le colonne della matrice rappresentano i valori predetti e le righe rappresentano i valori reali ottenuti dal processo di classificazione. L'elemento, ad esempio, sulla riga i e colonna j è il numero di casi in cui il modello ha classificato la classe i come classe j.

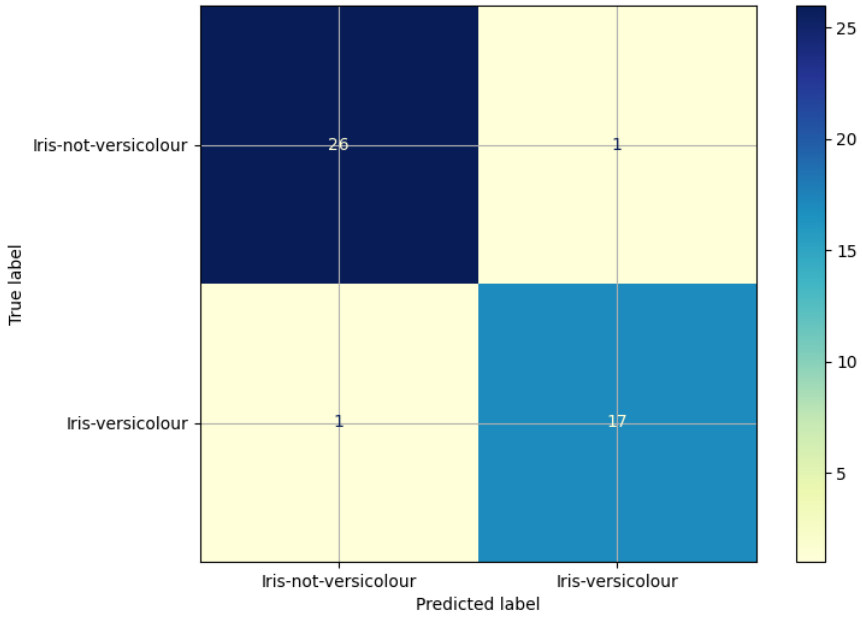
I valori mostrati nelle successive tabelle e grafici potrebbero variare leggermente.

La classificazione avviene su un test set costituito da 45 esempi (il 30% del dataset).

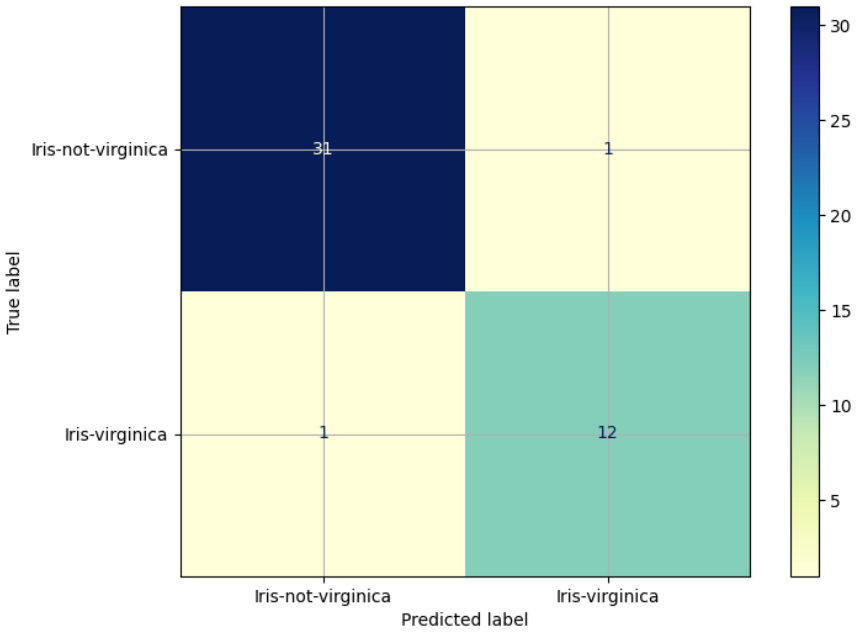
Per la classificazione eseguita sul dataset Iris_Setosa, Iris_Versicolour e Iris_Virginica i valori delle metriche prodotti dalla classificazione del modello Decision Tree sono i seguenti:



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Decision Trees	1.0	1.0	1.0	1.0	1.0	1.0



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Decision Trees	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9537037037037037	0.9619341563786008

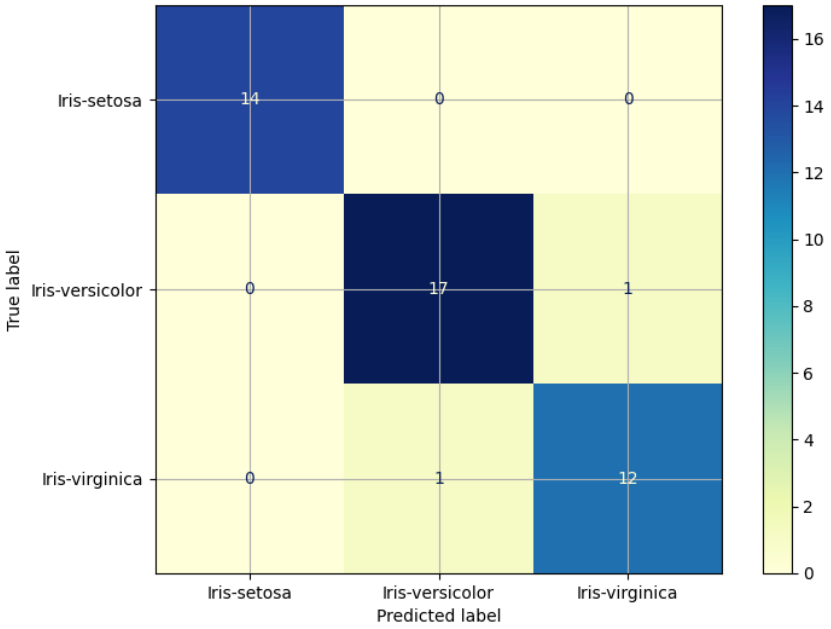


	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Decision Trees	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9459134615384616	0.9459134615384616

Come nella classificazione del modello KNN, il Decision Tree ottiene una classificazione perfetta durante la classificazione del dataset Iris_Setosa, ma ottiene circa il 95% per ogni metrica durante la classificazione dei dataset Iris_Versicolour e Iris_Virginica.

Questo risultato potrebbe variare con più esecuzioni della classificazione.

Per quanto riguarda la classificazione del dataset originale Iris, i risultati sono i seguenti:



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Decision Trees	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9558404558404558	0.9780966418697901

Il valore delle metriche accuracy, f1, precision e recall associate alla classificazione del dataset Iris è molto simile ai valori corrispondenti per la classificazione binaria effettuata sul dataset Iris_Versicolour; detto ciò, il valore delle metriche balanced accuracy e auc per questa classificazione sono migliorate circa del 0,2 % e 2% rispettivamente.

Random Forest

Sommario

La classificazione si esegue dividendo gli attributi all'interno del dataset in due gruppi: gli attributi di input di cui si conoscono sempre i valori e gli attributi di output su cui effettuare la classificazione.

Nel nostro caso gli attributi di input sono i seguenti: sepal length (lunghezza del sepalo), sepal width (larghezza del sepalo), petal length (lunghezza del petalo) e pedal width (larghezza del petalo).

L'attributo di output invece, nel nostro caso sarà la classe del fiore (setosa, versicolour e virginica nel caso di classificazione multi-classe e, ad esempio, setosa/non setosa nel caso di classificazione binaria).

Successivamente, il dataset viene diviso, tramite la procedura train_test_split in 4 set di istanze: X_train, X_test, y_train, y_test.

Una volta eseguito il processo di classificazione, si crea un file classification_statements, il quale contiene le asserzioni derivate dai risultati della classificazione, successivamente la KB legge questo file e ottiene, tramite l'azione Assert, le metriche relative alla classificazione effettuata dal modello Random Forest associate alla classe o classi su cui si è effettuata.

Strumenti utilizzati

La Random Forest è un modello composto da più alberi di decisione, quindi è un esempio di ensemble learning (nel quale si combinano le predizioni di più modelli), nel quale gli alberi di decisione sono la base e la predizione finale è ottenuta come media o voto delle predizioni singole.

L'idea è di addestrare un numero di alberi di decisione su parte del dataset e, per ogni esempio da classificare, ogni albero effettua una predizione; ottenendo la predizione finale per l'esempio aggregando le predizioni singole.

Decisioni di Progetto

Per implementare il modello della Random Forest Classifier abbiamo utilizzato il modulo RandomForestClassifier dalla libreria sklearn.ensemble e, per migliorare la sua efficacia abbiamo implementato la procedura K-Fold CrossValidation tramite la funzione GridSearchCV della libreria sklearn.model_selection.

RandomForestClassifier:
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

GridSearchCV:
https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Valutazione

Le metriche utilizzate per misurare l'efficacia della classificazione eseguita dalla Random Forest sono le stesse usate nella valutazione dei precedenti modelli, ovvero le seguenti: Accuracy, F1 Score, Precision, Recall, Balanced Accuracy e l'AUC Score.

Tutte le metriche utilizzate fanno parte del modulo sklearn.metrics: https://scikit-learn.org/stable/modules/model_evaluation.html

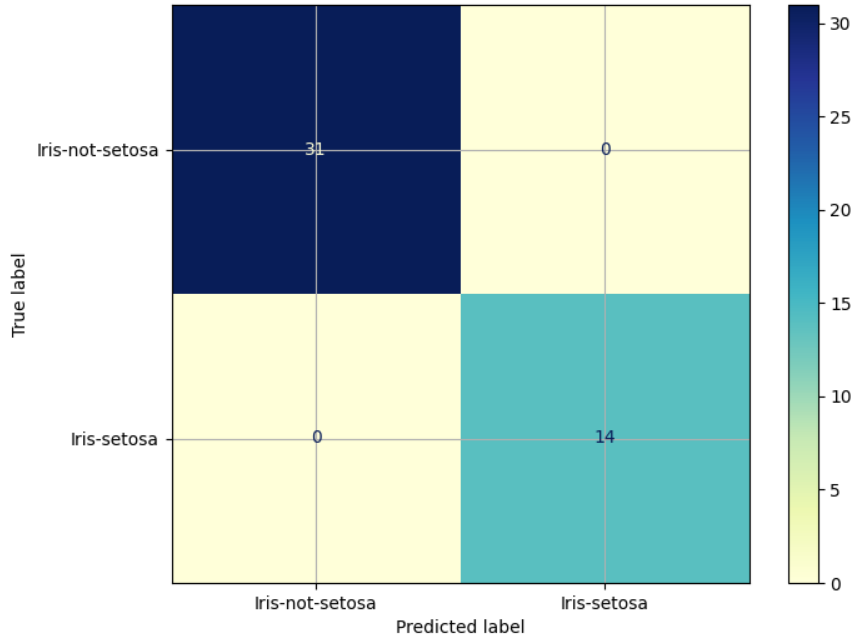
Aggiunti i valori delle metriche nella KB, esse posso essere oggetto di query eseguite dall'azione Ask.

Per ogni modello abbiamo costruito una matrice di confusione al fine di rappresentare l'accuratezza della classificazione. Le colonne della matrice rappresentano i valori predetti e le righe rappresentano i valori reali ottenuti dal processo di classificazione. L'elemento, ad esempio, sulla riga i e colonna j è il numero di casi in cui il modello ha classificato la classe i come classe j.

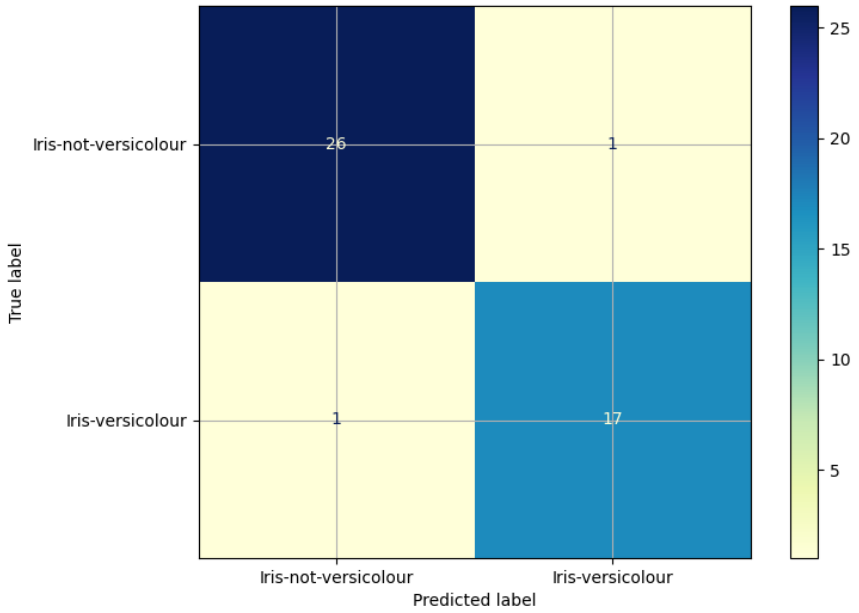
I valori mostrati nelle successive tabelle e grafici potrebbero variare leggermente.

La classificazione avviene su un test set costituito da 45 esempi (il 30% del dataset).

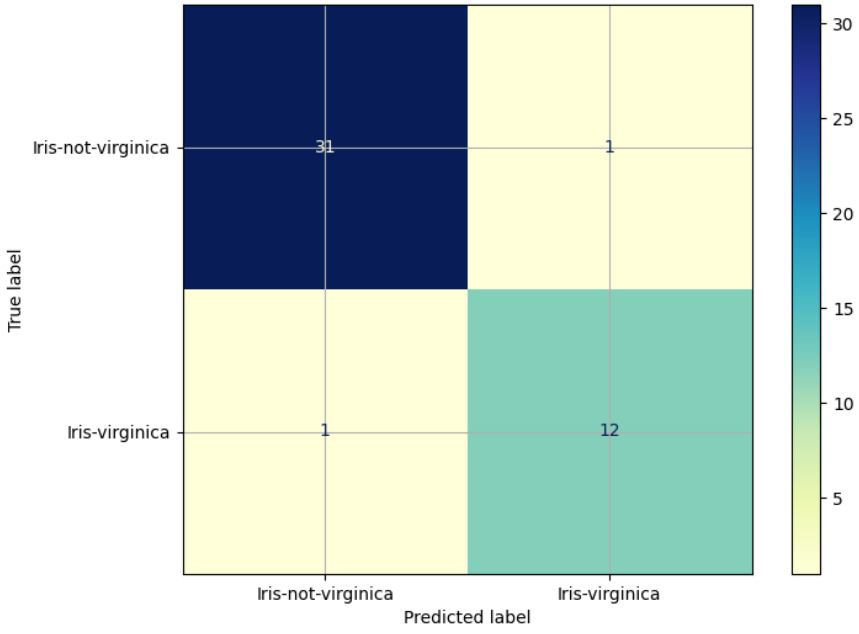
Per la classificazione eseguita sul dataset Iris_Setosa, Iris_Versicolour e Iris_Virginica i valori delle metriche prodotti dalla classificazione del modello Random Forest sono i seguenti:



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Random Forest	1.0	1.0	1.0	1.0	1.0	1.0



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Random Forest	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9537037037037037	0.9958847736625516

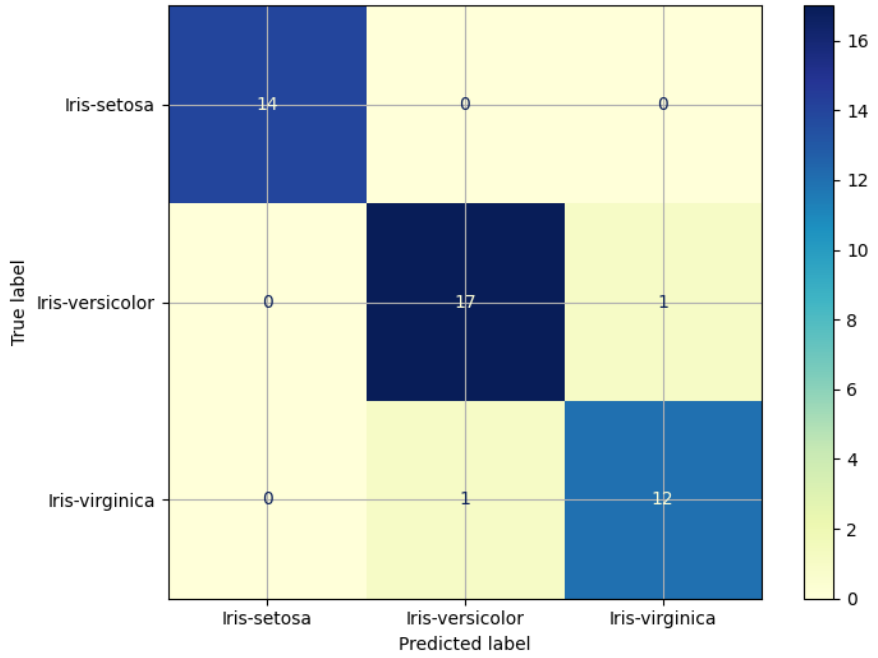


	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Random Forest	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9459134615384616	0.9939903846153846

Come nelle classificazioni eseguite dai modelli precedenti, tutte e tre le classificazioni binarie sono ottime, in particolare quella effettuata sul dataset Iris_Setosa. Le altre due hanno valori molto simili per quanto riguarda le metriche di accuracy, f1, precision e recall; invece, le metriche di balanced_accuracy e auc sono diverse circa del 8% e 2% rispettivamente.

Questo risultato potrebbe variare con più esecuzioni della classificazione.

Per quanto riguarda la classificazione del dataset originale Iris, i risultati sono i seguenti:



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Random Forest	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9558404558404558	0.9955385406774296

I valori delle metriche calcolate sulla classificazione del dataset Iris sono molto simili ai valori già osservati nella classificazione binaria dei file Iris_Versicolour e Iris_Virginica per quanto riguarda le metriche accuracy, f1, precision e recall. La balanced accuracy e la auc, invece, sono aumentati dello 0,2 % e 1,2 % rispettivamente a confronto della classificazione del dataset Iris_Virginica; rispetto alla classificazione del dataset Iris_Versicolour, solo la balanced accuracy è aumentata del 0,2 %.

Logistic Regression

Sommario

La classificazione si esegue dividendo gli attributi all'interno del dataset in due gruppi: gli attributi di input di cui si conoscono sempre i valori e gli attributi di output su cui effettuare la classificazione.

Nel nostro caso gli attributi di input sono i seguenti: sepal length (lunghezza del sepalo), sepal width (larghezza del sepalo), petal length (lunghezza del petalo) e pedal width (larghezza del petalo).

L'attributo di output invece, nel nostro caso sarà la classe del fiore (setosa, versicolour e virginica nel caso di classificazione multi-classe e, ad esempio, setosa/non setosa nel caso di classificazione binaria).

Successivamente, il dataset viene diviso, tramite la procedura train_test_split in 4 set di istanze: X_train, X_test, y_train, y_test.

Una volta eseguito il processo di classificazione, si crea un file classification_statements, il quale contiene le asserzioni derivate dai risultati della classificazione, successivamente la KB legge questo file e ottiene, tramite l'azione Assert, le metriche relative alla classificazione effettuata dal classificatore Logistic Regressor associate alla classe o classi su cui si è effettuata.

Strumenti utilizzati

Il Logistic Regression è un classificatore basato sulla regressione logistica, ovvero un modello di regressione nonlineare utilizzato per modellare la feature di output, solo se dicotomica (nel nostro caso la feature classe lo è).

L'obiettivo del modello è di stabilire la probabilità con cui un'osservazione può generare uno dei valori della feature di output in base alle feature da noi note (gli attributi del fiore nel nostro dataset)

Per più informazioni, tra cui i vari tipi di regressione logistica e confronto con la regressione lineare: https://www.ibm.com/it-it/topics/logistic-regression

Decisioni di Progetto

Per implementare il classificatore Logistic Regression abbiamo utilizzato il modulo LogisticRegression dalla libreria sklearn.linear_model e per normalizzare i valori all'interno del dataset scelto abbiamo utilizzato il modulo StandardScaler della libreria sklearn.preprocessing.

Infine, per migliorare l'efficacia della classificazione abbiamo implementato la procedura K-Fold CrossValidation tramite la funzione GridSearchCV della libreria sklearn.model_selection.

Logistic Regression:
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

StandardScaler:
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

GridSearchCV:
https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Valutazione

Le metriche utilizzate per misurare l'efficacia della classificazione eseguita dalla Logistic Regression sono le stesse usate nella valutazione dei precedenti modelli, ovvero le seguenti: Accuracy, F1 Score, Precision, Recall, Balanced Accuracy e l'AUC Score.

Tutte le metriche utilizzate fanno parte del modulo sklearn.metrics: https://scikit-learn.org/stable/modules/model_evaluation.html

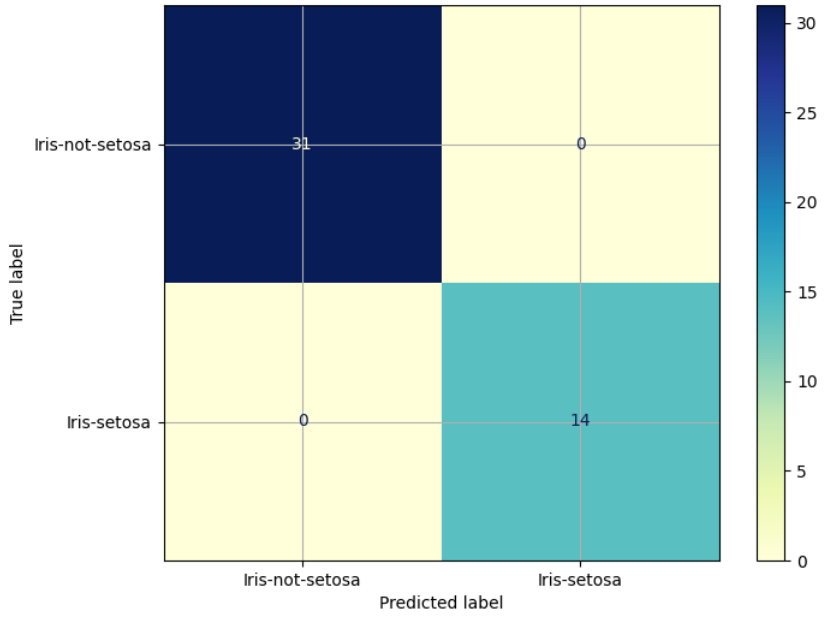
Aggiunti i valori delle metriche nella KB, esse posso essere oggetto di query eseguite dall'azione Ask.

Per ogni modello abbiamo costruito una matrice di confusione al fine di rappresentare l'accuratezza della classificazione. Le colonne della matrice rappresentano i valori predetti e le righe rappresentano i valori reali ottenuti dal processo di classificazione. L'elemento, ad esempio, sulla riga i e colonna j è il numero di casi in cui il modello ha classificato la classe i come classe j.

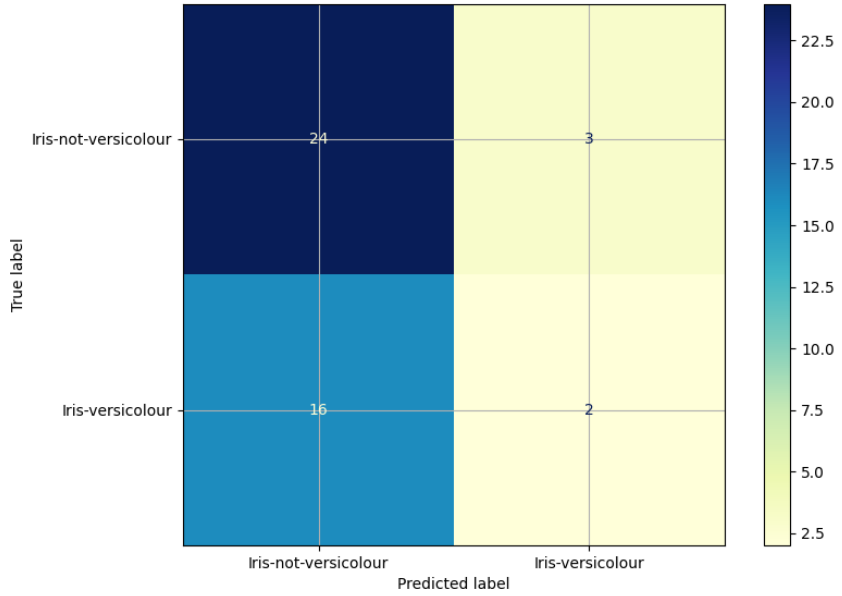
I valori mostrati nelle successive tabelle e grafici potrebbero variare leggermente.

La classificazione avviene su un test set costituito da 45 esempi (il 30% del dataset).

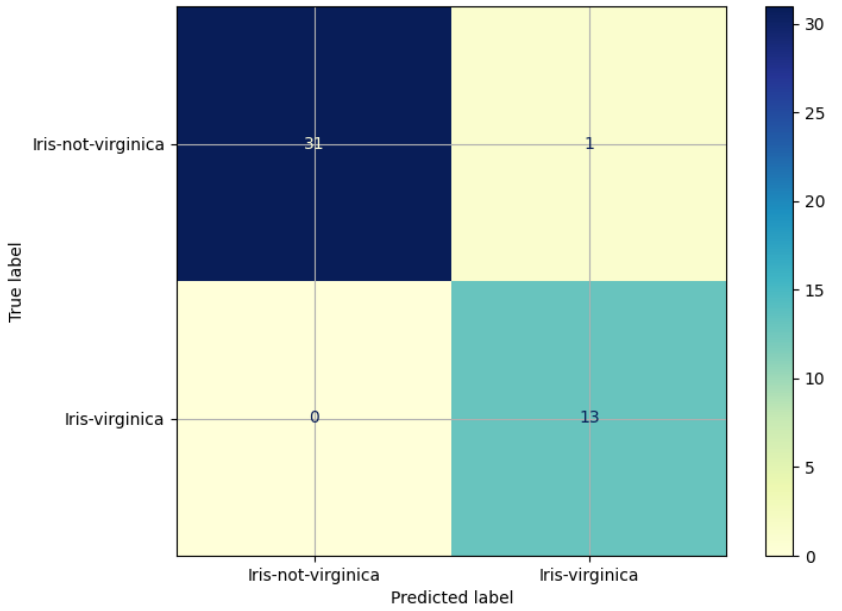
Per la classificazione eseguita sul dataset Iris_Setosa, Iris_Versicolour e Iris_Virginica i valori delle metriche prodotti dalla classificazione del modello Logistic Regression sono i seguenti:



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Logistic Regression	1.0	1.0	1.0	1.0	1.0	1.0



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Logistic Regression	0.5777777777777777	0.5777777777777777	0.5777777777777777	0.5777777777777777	0.5	0.7839506172839507

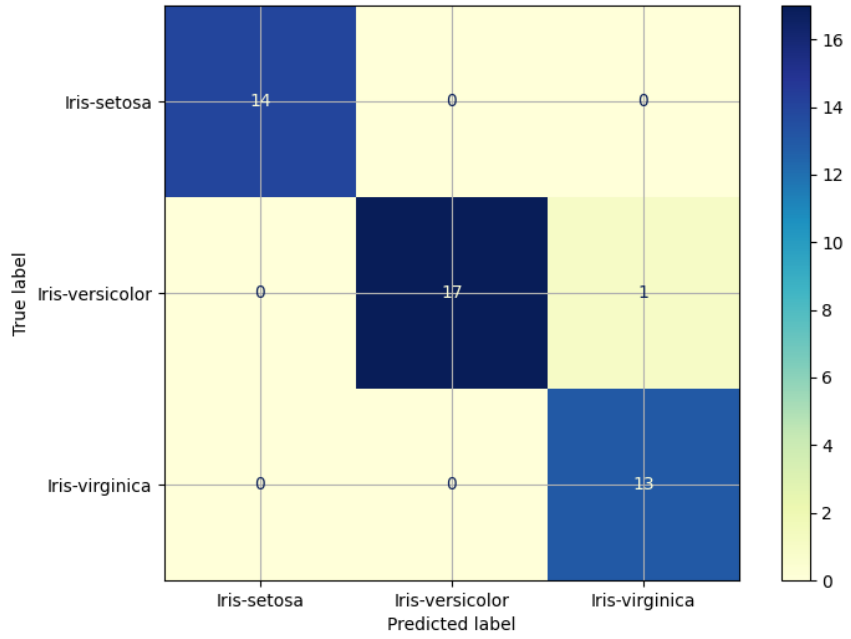


	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Logistic Regression	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.984375	1.0

A differenza delle classificazioni effettuate dai modelli precedenti, la classificazione effettuata sul dataset Iris_Setosa è ottima, ma le altre due hanno valori delle metriche molto diversi tra di loro. Nella classificazione eseguita sul dataset Iris_Versicolour, anche se eseguita più volte, si possono osservare valori diminuiti drasticamente (da circa 1,0 o 0,97 nelle altre due classificazioni per le metriche accuracy, f1, precision e recall a 0,57 per le stesse metriche; per i valori della balanced accuracy osserviamo una diminuzione da circa 1,0 e 0,98 a 0,5 e, infine, per i valori della auc riscontriamo una diminuzione da 1,0 a 0,78).

Questo risultato potrebbe variare con più esecuzioni della classificazione (anche se il comportamento anomalo osservato verificatosi durante la classificazione del dataset Iris_Versicolour si è ripetuto su più esecuzioni con risultati molto simili a quelli già mostrati).

Per quanto riguarda la classificazione del dataset originale Iris, i risultati sono i seguenti:



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Logistic Regression	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9814814814814815	1.0

Come possiamo osservare, i valori delle metriche associate alla classificazione del dataset Iris sono molto simili ai valori ottenuti durante la classificazione binaria del dataset Iris_Virginica.
Questa considerazione vale particolarmente per le metriche di accuracy, f1, precision, recall e auc, mentre la balanced accuracy è diminuita di circa 0,003, una differenza minima che non influisce sull'efficacia della classificazione e sulla considrazione fatta precedentemente.

Neural Network

Sommario

La classificazione si esegue dividendo gli attributi all'interno del dataset in due gruppi: gli attributi di input di cui si conoscono sempre i valori e gli attributi di output su cui effettuare la classificazione.

Nel nostro caso gli attributi di input sono i seguenti: sepal length (lunghezza del sepalo), sepal width (larghezza del sepalo), petal length (lunghezza del petalo) e pedal width (larghezza del petalo).

L'attributo di output invece, nel nostro caso sarà la classe del fiore (setosa, versicolour e virginica nel caso di classificazione multi-classe e, ad esempio, setosa/non setosa nel caso di classificazione binaria).

Successivamente, il dataset viene diviso, tramite la procedura train_test_split in 4 set di istanze: X_train, X_test, y_train, y_test.

Una volta eseguito il processo di classificazione, si crea un file classification_statements, il quale contiene le asserzioni derivate dai risultati della classificazione, successivamente la KB legge questo file e ottiene, tramite l'azione Assert, le metriche relative alla classificazione effettuata dalla neural network (rete neurale) associate alla classe o classi su cui si è effettuata.

Strumenti utilizzati

Una rete neurale è un modello computazionale non-lineare costituito da unità artificiali ispirate dai neuroni presenti in una rete neurale biologica; questo tipo di modelli viene spesso utilizzato per tentare di risolvere problemi ingegneristici di intelligenza artificiale in vari ambiti tecnologici (in elettronica, informatica e altre discipline...).

L'architettura della rete neurale è riconducibile a tre tipi di gruppi composti dalle unità artificiali della rete. Questi tre gruppi sono: unità di input, unità nascoste e unità di output.

Le unità di input alimentano lo stato o gli strati nascosti di unità (ogni unità prevede una funzione semplice che ha l'obiettivo di combinare output di unità di strati inferiori) che, a loro volta, alimenteranno le predizioni dello strato di output.

La rete neurale da noi utilizzata è un classificatore percettrone a più strati che ottimizza la funzione log-loss usando l'ottimizzatore Limited-memory BFGS (nella famiglia dei metodi di Quasi-Newton).

Decisioni di Progetto

Per implementare il classificatore precettrone a più strati abbiamo utilizzato il modulo MLPClassifier dalla libreria sklearn.neural_network. Spesso questo tipo di classificatore viene utilizzato per dataset con molti elementi, quindi nel nostro caso abbiamo scelto i parametri con i quali creare il modello in modo da rendere il suo utilizzo adeguato anche sul nostro dataset di 150 esempi.

Alcuni dei parametri più importanti da noi scelti sono i seguenti:
hidden_layer_sizes: (4,4,4) --> indica il numero di unità in ciascun strato nascosto (nel nostro caso 3)
max_iter: [2000, 2100] --> indica il numero massimo di iterazioni (utile diminuire il suo valore data la dimensione ridotta del nostro dataset e potrebbe richiedere modifica a seconda della macchina sulla quale viene eseguito questo progetto)

solver: [lbfgs] --> risolutore raccomandato per dataset di dimensioni ridotte poichè converge più rapidamente e preforma meglio del risolutore predefinito.

Infine, per aumentare la sua efficacia, abbiamo implementato la procedura K-Fold CrossValidation tramite la funzione GridSearchCV della libreria sklearn.model_selection.

MLPClassifier:
https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

GridSearchCV:
https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Valutazione

Le metriche utilizzate per misurare l'efficacia della classificazione eseguita dalla rete neurale sono le stesse usate nella valutazione dei precedenti modelli, ovvero le seguenti: Accuracy, F1 Score, Precision, Recall, Balanced Accuracy e l'AUC Score.

Tutte le metriche utilizzate fanno parte del modulo sklearn.metrics:
https://scikit-learn.org/stable/modules/model_evaluation.html

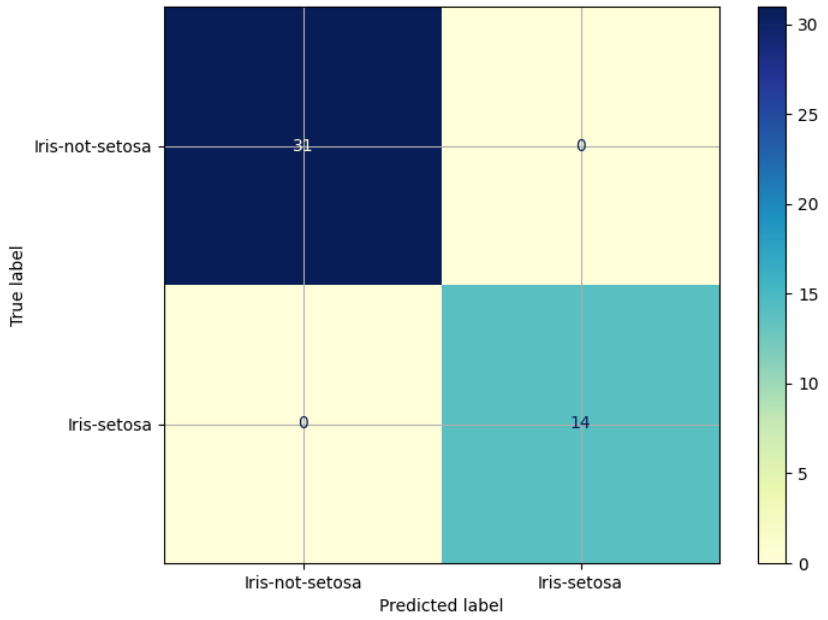
Aggiunti i valori delle metriche nella KB, esse posso essere oggetto di query eseguite dall'azione Ask.

Per ogni modello abbiamo costruito una matrice di confusione al fine di rappresentare l'accuratezza della classificazione. Le colonne della matrice rappresentano i valori predetti e le righe rappresentano i valori reali ottenuti dal processo di classificazione. L'elemento, ad esempio, sulla riga i e colonna j è il numero di casi in cui il modello ha classificato la classe i come classe j.

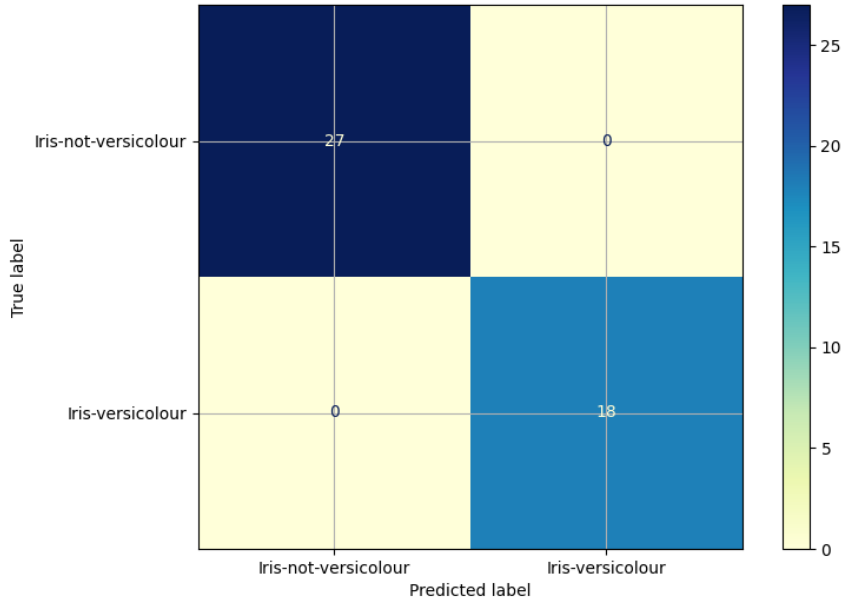
I valori mostrati nelle successive tabelle e grafici potrebbero variare leggermente.

La classificazione avviene su un test set costituito da 45 esempi (il 30% del dataset).

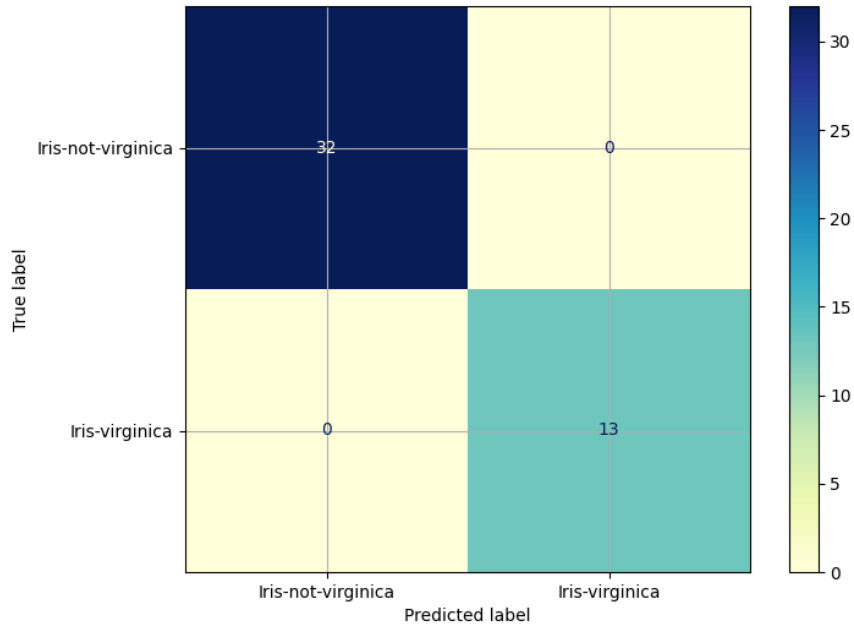
Per la classificazione eseguita sul dataset Iris_Setosa, Iris_Versicolour e Iris_Virginica i valori delle metriche prodotti dalla classificazione del modello Neural Network sono i seguenti:



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Neural Network	1.0	1.0	1.0	1.0	1.0	1.0



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Neural Network	1.0	1.0	1.0	1.0	1.0	1.0

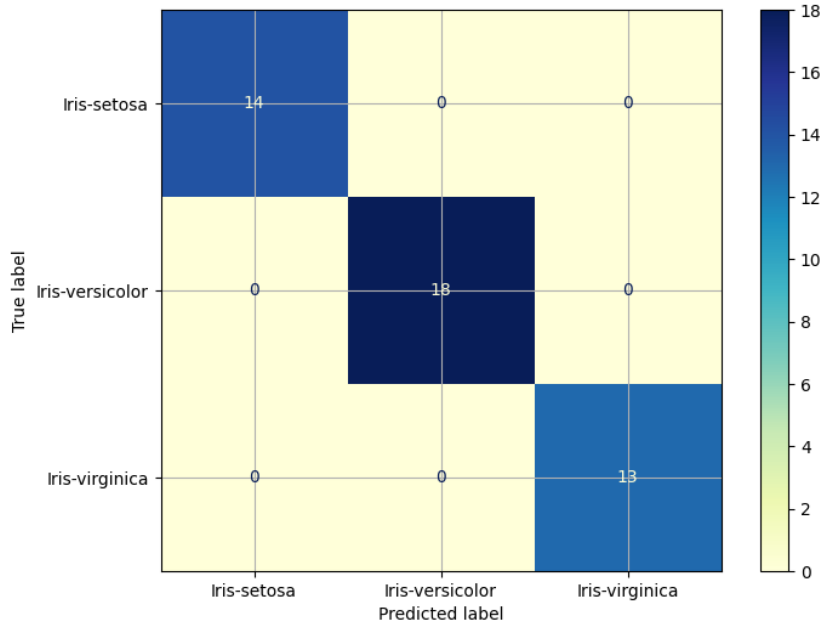


	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Neural Network	1.0	1.0	1.0	1.0	1.0	1.0

Il modello Neural Network è l'unico modello tra quelli da noi implementati in grado di fornire dei valori ottimi e consistenti per tutte le classificazione binarie effettuate sui dataset Iris_Setosa, Iris_Versicolour e Iris_Virginica.

Si noti che, durante l'esecuzione della classificazione effettuata da questo modello, si potrebbero incontrare errori riguardanti il numero delle iterazione e la convergenza. Dopo più test, effettuati su diverse macchine, siamo arrivati alla conclusione che il valore ottimale di questi parametri deve essere modificato dall'utente, in quanto dipendono dall'hardware della macchina su cui viene eseguito il codice. Detto ciò, questo errore non influenza in maniera evidente i risultati da noi ottenuti.

Per quanto riguarda la classificazione del dataset originale Iris, i risultati sono i seguenti:



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Neural Network	1.0	1.0	1.0	1.0	1.0	1.0

Come abbiamo già constatato nelle classificazioni binarie eseguite da questo modello precedentemente discusse, la rete neurale riesce ad ottenere una classificazione impeccabile anche nel caso in cui la classificazione sia effettuata su più classi (Setosa, Versicolour e Virginica).

Questi valori delle metriche è consistente su più esecuzioni della classificazione effettuata dal modello Neural Network.

Naive Bayes

Sommario

La classificazione si esegue dividendo gli attributi all'interno del dataset in due gruppi: gli attributi di input di cui si conoscono sempre i valori e gli attributi di output su cui effettuare la classificazione.

Nel nostro caso gli attributi di input sono i seguenti: sepal length (lunghezza del sepalo), sepal width (larghezza del sepalo), petal length (lunghezza del petalo) e pedal width (larghezza del petalo).

L'attributo di output invece, nel nostro caso sarà la classe del fiore (setosa, versicolour e virginica nel caso di classificazione multi-classe e, ad esempio, setosa/non setosa nel caso di classificazione binaria).

Successivamente, il dataset viene diviso, tramite la procedura train_test_split in 4 set di istanze: X_train, X_test, y_train, y_test.

Una volta eseguito il processo di classificazione, si crea un file classification_statements, il quale contiene le asserzioni derivate dai risultati della classificazione, successivamente la KB legge questo file e ottiene, tramite l'azione Assert, le metriche relative alla classificazione effettuata dal classificatore Naive Bayes associate alla classe o classi su cui si è effettuata.

Strumenti utilizzati

Il classificatore Naive Bayes è un modello di classificazione probabilitsico il cui obiettivo è imparare la dipendenza delle feature della classe e utilizzare la rappresentazione così costruita per predire la classificazione di nuovi esempi.

Per utilizzare questo modello è necessario assumere l'indipendenza condizionata tra le feature di input data la feature di output, assunzione verificata nel nostro caso.

Decisioni di Progetto

Per implementare il classificatore Naive Bayes abbiamo utilizzato il modulo CategoricalNB dalla libreria sklearn.naive_bayes. In questa libreria sono presenti più implementazione di algoritmi di Naive Bayes tra cui: BernulliNB, ComplementNB, GaussianNB e MultinomialNB; tra le citate abbiamo provato ad utilizzare per la classificazione la BernulliNB, la GaussianNB e la MultinomialNB a parte la CategoricalNB e abbiamo riscontrato che otteniamo le migliori metriche con la CategoricalNB, soprattutto a causa del tipo di valori presenti nelle feature di input del dataset IRIS.

Infine, per migliorare la sua efficacia abbiamo implementato la procedura K-Fold CrossValidation tramite la funzione GridSearchCV della libreria sklearn.model_selection.

CategoricalNB:
https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.CategoricalNB.html

GridSearchCV:
https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Valutazione

Le metriche utilizzate per misurare l'efficacia della classificazione eseguita dal classificatore Naive Bayes sono le stesse usate nella valutazione dei precedenti modelli, ovvero le seguenti: Accuracy, F1 Score, Precision, Recall, Balanced Accuracy e l'AUC Score.

Tutte le metriche utilizzate fanno parte del modulo sklearn.metrics: https://scikit-learn.org/stable/modules/model_evaluation.html

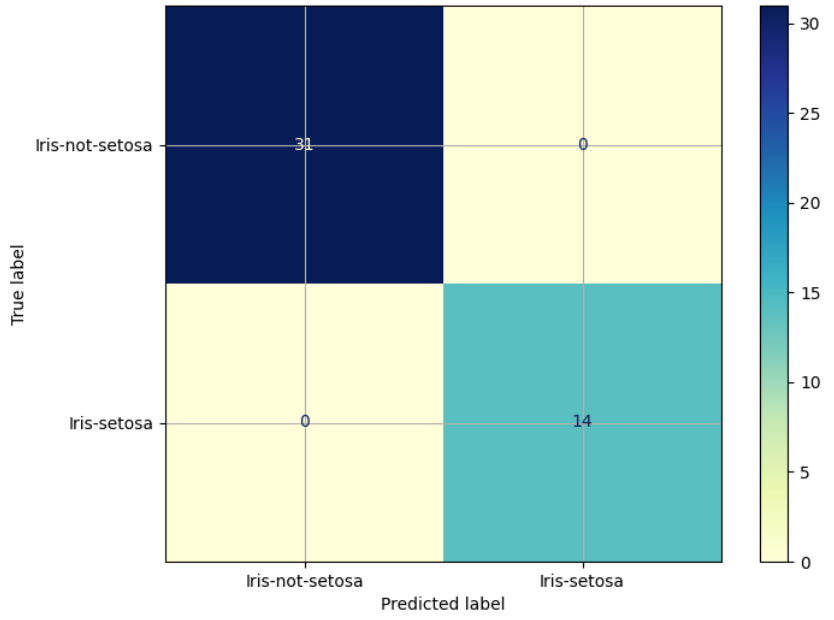
Aggiunti i valori delle metriche nella KB, esse posso essere oggetto di query eseguite dall'azione Ask.

Per ogni modello abbiamo costruito una matrice di confusione al fine di rappresentare l'accuratezza della classificazione. Le colonne della matrice rappresentano i valori predetti e le righe rappresentano i valori reali ottenuti dal processo di classificazione. L'elemento, ad esempio, sulla riga i e colonna j è il numero di casi in cui il modello ha classificato la classe i come classe j.

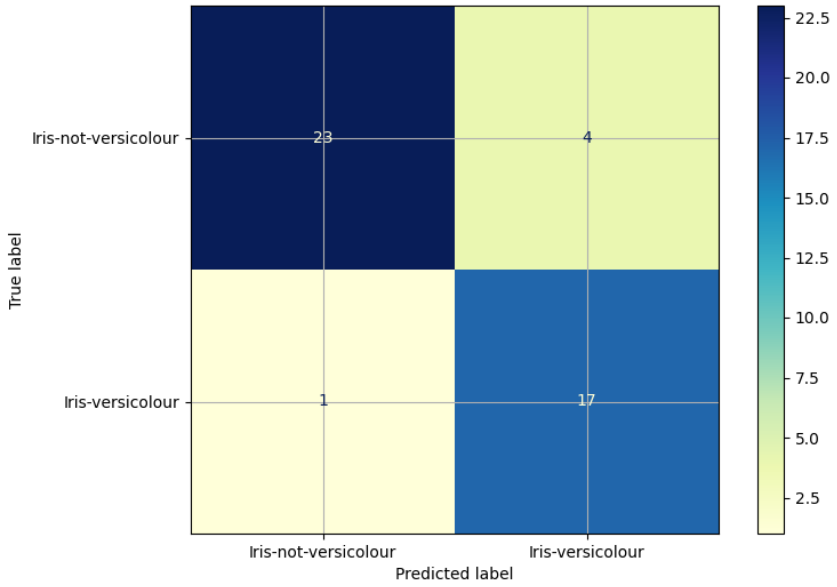
I valori mostrati nelle successive tabelle e grafici potrebbero variare leggermente.

La classificazione avviene su un test set costituito da 45 esempi (il 30% del dataset).

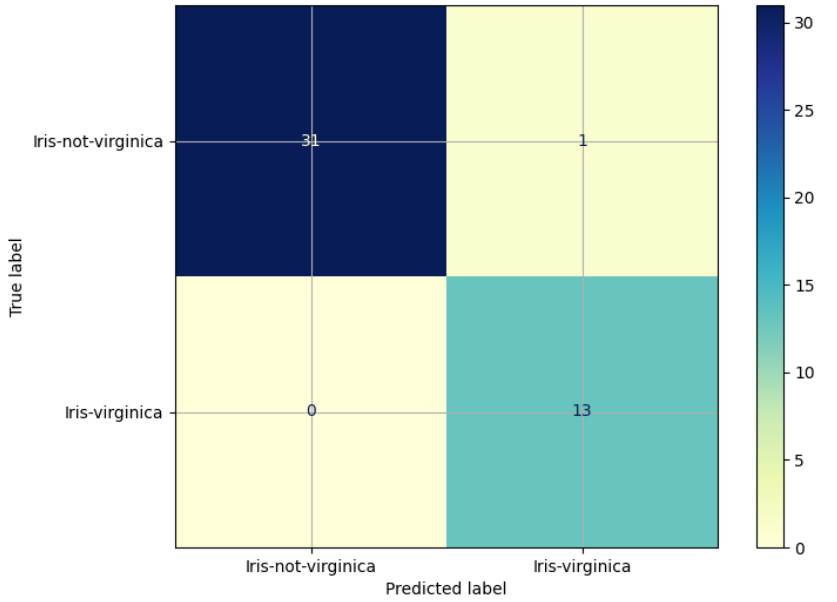
Per la classificazione eseguita sul dataset Iris_Setosa, Iris_Versicolour e Iris_Virginica i valori delle metriche prodotti dalla classificazione del modello Naive Bayes sono i seguenti:



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Naive Bayes	1.0	1.0	1.0	1.0	1.0	1.0



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Naive Bayes	0.8888888888888888	0.8888888888888888	0.8888888888888888	0.8888888888888888	0.8981481481481481	0.9917695473251028



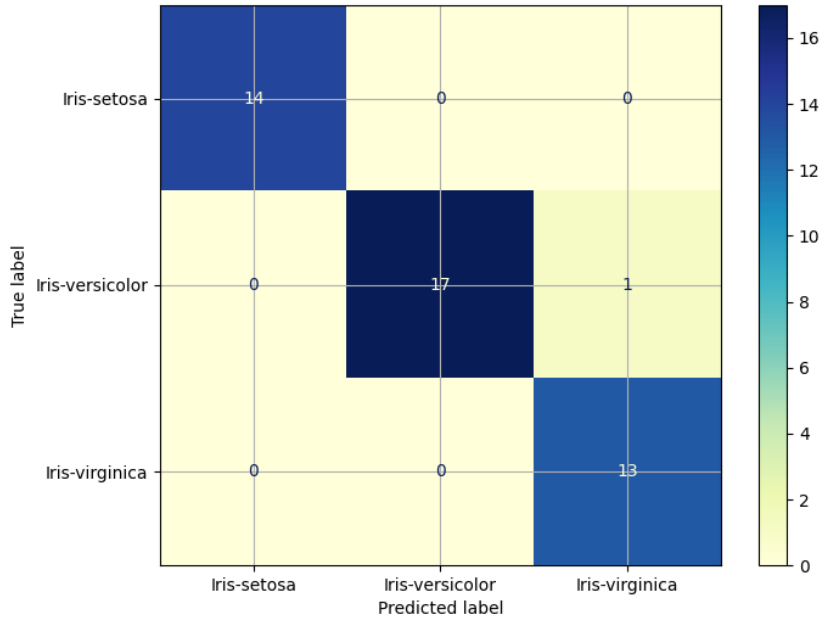
	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Naive Bayes	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.984375	1.0

Come riscontrato nei risultati delle classificazioni eseguita dai modelli precedenti, tutte e tre le classificazioni binarie sono buone, in particolare quella effettuata sul dataset Iris_Setosa.
Per le metriche accuracy, f1, precision e recall la classificazione sul dataset Iris_Versicolour ottiene valori dimiuiti del 12% rispetto ai risultati sulla classificazione di Iris_Setosa e del 9% rispetto ai risultati sulla classificazione di Iris_Virginica, la quale ottiene valori diminuiti di circa 3% rispetto ai risultati sulla classificazione di Iris_Setosa.
Per la metrica balanced accuracy osserviamo il valore più alto ottenuto dalla classificazione su

Iris_Setosa, un valore diminuito del 2% dalla classificazione su Iris_Virginica e un valore diminuito dell'11% (rispetto alla classificazione su Iris_Setosa) dalla classificazione su Iris_Versicolour.

Questo risultato potrebbe variare con più esecuzioni della classificazione.

Per quanto riguarda la classificazione del dataset originale Iris, i risultati sono i seguenti:



	accuracy	f1_score	precision	recall	balanced_accuracy	auc
Naive Bayes	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9814814814814815	0.9940513875699061

Come osservato nei risultati ottenuti dalle classificazione eseguite dai precedenti modelli, i valori delle metriche accuracy, f1, precision e recall sono molto simili ai valori ottenuti dalla classificazione binaria sul dataset Iris_Virginica.
Il valore della metrica balanced accuracy è diminuito del 0,3% rispetto al valore ottenuto dalla classificazione sopra citata e il valore della metrica auc è diminuito del 0,7% rispetto ai valori ottenuti dalle classificazioni binarie sui dataset Iris_Setosa e Iris_Virginica ed è aumentato del 0,3% rispetto al valore ottenuto dalla classificazione binaria sul dataset Iris_Versicolour.

K-Means

Sommario

Essendo un modello di apprendimento non supervisionato, la classificazione si esegue inizialmente applicando al dataset uno scaler (nello specifico il MinMaxScaler) per normalizzare i valori all'interno del dataset e, successivamente, creando il modello indicando, tra i suoi parametri, il numero delle classi a cui appartengono gli esempi presenti all'interno del dataset. Infine, si effettua la fase di addestramento del modello costruito sul dataset normalizzato indicando le feature presenti al suo interno.

Una volta eseguito il processo di classificazione, si crea un file classification_statements, il quale contiene le asserzioni derivate dai risultati della classificazione, successivamente la KB legge questo file e ottiene, tramite l'azione Assert, lo score relativo alla classificazione effettuata dall'algoritmo K-Means associate alle classi su cui si è effettuata.

Strumenti utilizzati

La K-Means è un algoritmo per l'hard clustering che ottiene in input esempi di apprendimento e un numero di classi a cui appartengono gli esempi e fornisce in output uno score, ovvero l'assegnazione degli esempi all'interno del dataset alle classi segnalate durante la creazione del modello.

L'obiettivo del modello è, dato il training set con le sue feature di input e il relativo valore di ogni feature per ogni esempio del dataset, classificare gli esempi costruendo una funzione class che associa ad ogni esempio una classe.

L'algoritmo converge verso un minimo locale, migliorando iterativamente l'errore quadratico, il quale tende a ridursi in quanto c'è solo un numero finito di possibili riassegnazioni.

Decisioni di Progetto

Per implementare l'algoritmo K-Means abbiamo utilizzato il modulo KMeans dalla libreria sklearn.cluster e per normalizzare il dataset abbiamo utilizzato il modulo MinMaxScaler della libreria sklearn.preprocessing.

Per migliorare la sua efficacia abbiamo implementato la procedura K-Fold CrossValidation tramite la funzione GridSearchCV della libreria sklearn.model_selection.

K-Means:
https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

MinMaxScaler:
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

GridSearchCV:
https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Valutazione

La metrica utilizzata per misurare l'efficacia della classificazione eseguita dall'algoritmo K-Means è lo score: quanti esempi la funzione class dell'algoritmo ha assegnato a ciascuna classe (cluster) del dataset.

Aggiunto lo score della K-Means, esso può essere oggetto di query eseguite dall'azione Ask.

Per il valore dello score della k-means abbiamo deciso semplicemente di stamparlo nel terminale, quindi, per ogni classificazione, nel terminale verrà stampata una piccola tabella contenente 2 colonne: una colonna chiamata cluster, le cui righe contengono il numero del cluster a cui sono stati assegnati gli esempi e una colonna le cui riche contengono il numero degli esempi assegnati al rispettivo cluster (ovvero una delle classi del dataset selezionato).

La classificazione avviene su tutto il dataset, quindi 150 esempi.

I risultati ottenuti dalla classificazione binaria (quindi con 2 cluster) effettuata dal modello K-Means sui dataset Iris_Setosa, Iris_Versicolour e Iris_Virginica sono i seguenti:

```
 cluster
1    100
0     50
```

```
 cluster
0    100
1     50
```

```
 cluster
0    100
1     50
```

Osservando i risultati di queste classificazioni binarii, ogni classificazione ha riportato uno score perfetto; infatti, nei dataset binari da noi creati, modificato il dataset originale, sono presenti 50 esempi apparteneti alla classe omonima al nome del file e 100 esempi non appartenenti a quella classe.

Il risultato della classificazione effettuata dal modello K-Means sul dataset Iris è il seguente:

```
 cluster
2    61
1    50
0    39
```

Questa classificazione viene effettuata su un dataset contenente, come già indicato precedentemente, 50 esempi per ciascuna delle 3 classi al suo interno. Da ciò possiamo dedurre che il K-Means non eseguirà una classificazione perfetta, infatti ha associato il numero giusto di esempi, cioè 50 al cluster 1, ma al cluster 0 ha associato solo 39 esempi, associando 11 esempi in più al cluster 2.

Conclusioni

Per confrontare i valori ottenuti da tutte le classificazioni precedentemente mostrate e per discuterne riportiamo di seguente le tabelle contenenti le metriche (accuracy, f1, precision, recall, balanced accuracy e AUC) e la tabella contenente il k-means score per ciascuna delle classificazioni binarie eseguite sui dataset modificati (Iris_Setosa, Iris_Versicolour, Iris_Virginica) e della classificazione effettuata sul dataset originale Iris.

Risultati classificazione binaria dataset Iris_Setosa:

	accuracy	f1_score	precision	recall	balanced_accuracy	auc
K Nearest Neighbors	1.0	1.0	1.0	1.0	1.0	1.0
Decision Trees	1.0	1.0	1.0	1.0	1.0	1.0
Random Forest	1.0	1.0	1.0	1.0	1.0	1.0
Naive Bayes	1.0	1.0	1.0	1.0	1.0	1.0
Neural Network	1.0	1.0	1.0	1.0	1.0	1.0
Logistic Regression	1.0	1.0	1.0	1.0	1.0	1.0

```
 cluster
1     100
0      50
```

Risultati classificazione binaria dataset Iris_Versicolour:

	accuracy	f1_score	precision	recall	balanced_accuracy	auc
K Nearest Neighbors	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9814814814814814	0.9958847736625515
Decision Trees	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9537037037037037	0.9619341563786008
Random Forest	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9537037037037037	0.9958847736625516
Naive Bayes	0.8888888888888888	0.8888888888888888	0.8888888888888888	0.8888888888888888	0.8981481481481481	0.9917695473251028
Neural Network	1.0	1.0	1.0	1.0	1.0	1.0
Logistic Regression	0.5777777777777777	0.5777777777777777	0.5777777777777777	0.5777777777777777	0.5	0.7839506172839507

```
 cluster
0     100
1      50
```

Risultati classificazione binaria dataset Iris_Virginica:

	accuracy	f1_score	precision	recall	balanced_accuracy	auc
K Nearest Neighbors	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9615384615384616	0.9987980769230769
Decision Trees	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9459134615384616	0.9459134615384616
Random Forest	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9459134615384616	0.9939903846153846
Naive Bayes	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.984375	1.0
Neural Network	1.0	1.0	1.0	1.0	1.0	1.0
Logistic Regression	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.984375	1.0

```
 cluster
0     100
1      50
```

Risultati classificazione dataset Iris:

	accuracy	f1_score	precision	recall	balanced_accuracy	auc
K Nearest Neighbors	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9743589743589745	0.9970256937849532
Decision Trees	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9558404558404558	0.9780966418697901
Random Forest	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9555555555555556	0.9558404558404558	0.9955385406774296
Naive Bayes	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9814814814814815	0.9940513875699061
Neural Network	1.0	1.0	1.0	1.0	1.0	1.0
Logistic Regression	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9777777777777777	0.9814814814814815	1.0

```
 cluster
2    61
1    50
0    39
```

Le classificazioni binaria hanno risultati molto interessanti; la classificazione su Iris_Setosa ottiene valori massimi per ogni modello eseguito ma le altre due classificazioni ottengono risultati molto diversi.

I risultati ottenuti dalle classificazioni binarie sui dataset Iris_Versicolour e Iris_Virginica effettuate da tutti i modelli eccetto logistic regression e naive bayes sono molto simili tra di loro. La logistic regression in particolare, specie sul dataset Iris_Versicolour ottiene un valore diminuto drasticamente su tutte le metriche di valutazione rispetto ai risultati ottenuti dalle altre esecuzione dello stesso modello su altre classificazioni binarie.

L'unico modello consistente in tutte le sue applicazioni, sia su classificazioni binarie sia in classificazioni multi-classe, è la neural network (rete neurale), la quale ottiene valori perfetti in ogni classificazione su tutte le metriche.

In generale, tralasciando questi comportamenti particolari, i valori ottenuti dagli altri modelli, specialmente nelle metriche accuracy, f1, precision e recall, sono molto simili tra la classificazione multi-classe e le classficazioni binarie effettuate sui dataset Iris_Versicolour e Iris_Virginica.

Per la metrica auc possiamo osservare che essa varia generalmente molto poco (nell'ordine di circa 0,05) per tutti i modelli eccetto per i decision tree, dove i valori variano dal 2% al 4%.

Infine, per quanto riguarda la metrica balanced accuracy, essa varia soprattutto quando calcolata sulle classificazioni eseguite dai modelli knn, decision trees e random forest. I valori possono variare dall' 1% al 2% quando si confrontano i valori ottenuti tra le classificazioni binarie su Iris_Versicolour e Iris_Virginica; il valore però aumenta del 9% (da 0,89 a 0,98) quando si confornano i risultati ottenuti dalla classificazione eseguita dal modello naive bayes sui dataset citati.

Tra i modelli che volevamo implementare ma non siamo riusciti, sia per motivi di tempo, sia perchè abbiamo riscontrato alcuni problemi (apprendimento da parte della KB da noi utilizzata, problemi di calcolo della probabilità condizionata ed altro) erano presenti la Belief Network Bayesiana (BBN) e la implementazione dei CSP.

Riferimenti Bibliografici

David L. Poole, Alan K. Mackworth Artificial Intelligence: Foundations of Computational Agents, 2nd Edition, Cambridge University Press 2017