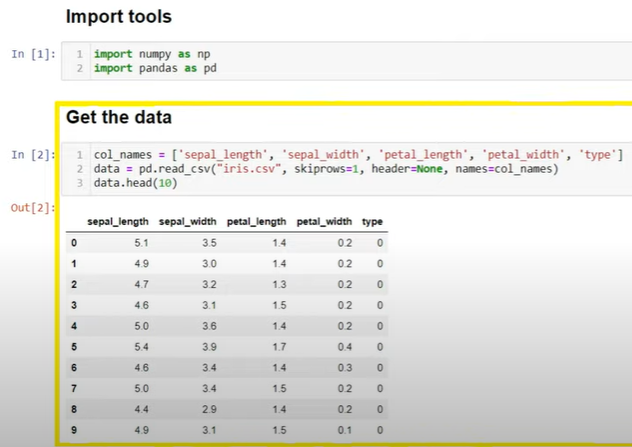
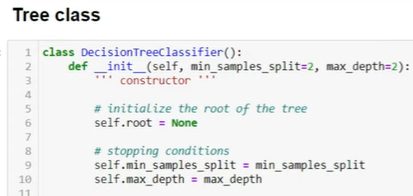
**IMPLEMENTAZIONE DECISION TREE ON JUPYTER:**



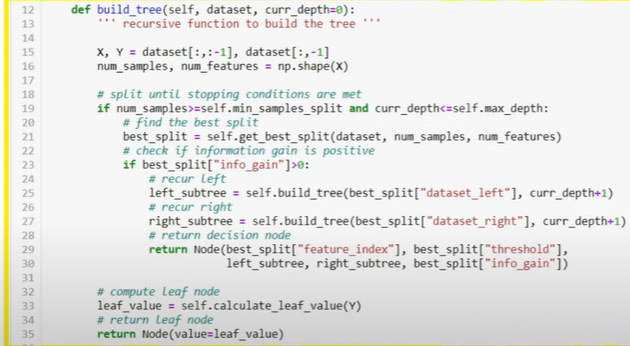
La classe Node descrive la base per i nostri alberi decisionali e implementa il costruttore di ogni oggetto node.

Graphical user interface, text, application, email

Description automatically generated

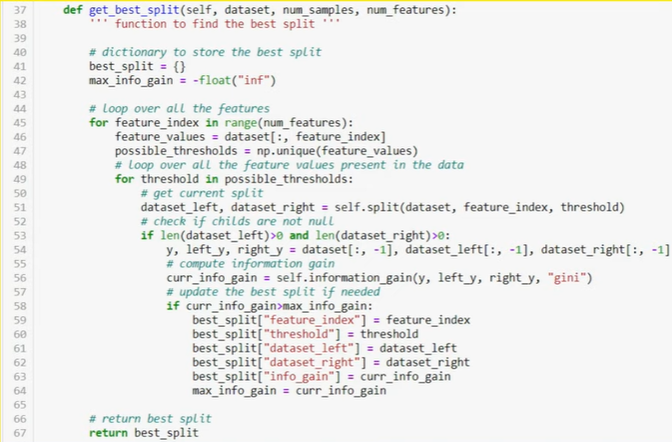


Questa prima funzione è il costruttore del nostro albero di decisione. Le 2 condizione di stop nella costruzione dell’albero sono o il #minimo di esempi contenuti in un nodo dopo uno split oppure la massima profondità che può raggiungere l’albero.



Questa è la più importante funzione che permetterà di costruire l’albero per ricorsione, la prima cosa che fa (line 15) divide il dataset passato in input in due insiemi le features e il target. In seguito, vengono estratti i numeri degli esempi e i numeri delle features con il metodo della libreria numpy np.shape(X).

Successivamente andiamo a fornire le condizioni di stop per la ricorsione(line18)all’ interno troviamo la chiamata alla stessa funzione per costruire il ramo destro e poi quello sinistro, e abbiamo la funzione get\_best\_split() che permetterà a seconda dei criteri dati per le condizioni di stop e tendono conto del Information Gain scegliere il modello di albero con i migliori test sui nodi interni.



In questa funzione realizziamo la scelta del migliore split per ogni test da eseguire

Text

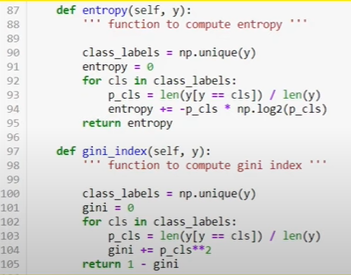
Description automatically generated

La funzione split() è quella che andrà a dividere l’insieme degli esempi presenti in un nodo padre e gli ripartirà a seconda del test e del IG nei due sotto insiemi da assegnare ai nodi figli.

Text

Description automatically generated

La funzione information\_gain() permetterà di calcolare IG per potere effettuare lo split.



Queste 2 funzioni calcolano rispettivamente la prima l’entropia in un nodo e la seconda l’indice di Gini.

Text

Description automatically generated with medium confidence

Questa funzione calculate\_leaf\_value() si occupa di calcolare quale che sia la classe maggioritaria per quel nodo foglia.

Text, application

Description automatically generated

Questa funzione servirà a stampare l’albero per averne una rappresentazione schematizzata.

Text

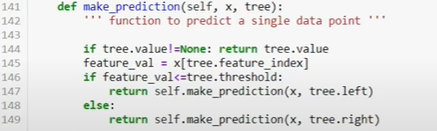
Description automatically generated

La funzione fit() servirà per addestrare il modello dell’albero decisionale

Graphical user interface, text

Description automatically generated

Questa funzione andrà ad effettuare le predizioni sui futuri dati passati al modello.



Questa è una funzione di ausilio per la precedente.

Ora una volta visto il codice che ci ha permesso di costruire il nostro albero decisionale da zero, vediamo il codice che permetterà di andare ad addestrare il nostro modello, la prima cosa sarà quella di andare a dividere il data set per creare training set e test set

Graphical user interface, text

Description automatically generated

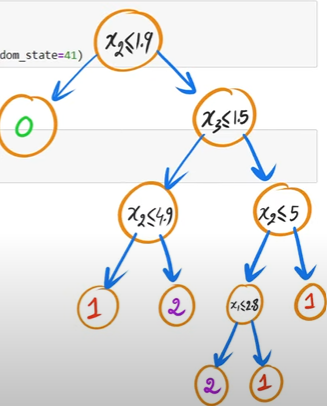
Successivamente diamo in pasto il training set

Graphical user interface, text, application

Description automatically generated

Infine andiamo a printare l’albero per vedere quello che ne risulta

Text

Description automatically generated 

Infine, andiamo a valutare il nostro modello su con il test set

Text

Description automatically generated

Calcoliamo l’accurancy con la funzione di metrica importata da sklearn.