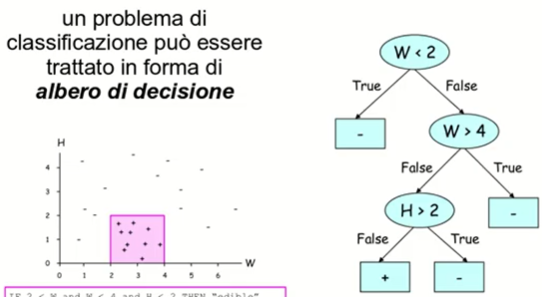
**ALBERI DI DECISIONE**

Vediamo adesso un problema di classificazione, i problemi di classificazione possono essere trattati con gli alberi decisionali.

L’albero decisionale può essere visto come una serie di test uno dopo l’altro.

****



**DECISION TREE CLASSIFICATION**

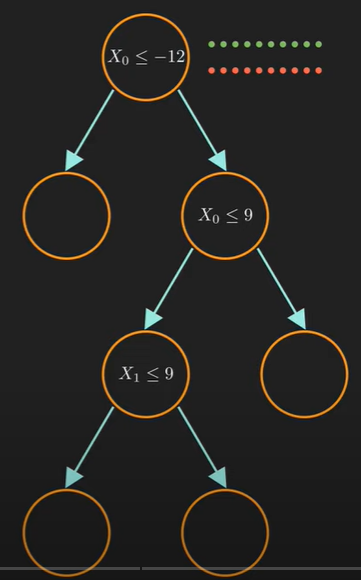
Partiamo da un data set che andiamo a rappresentare come uno scatter plot, e supponiamo che gli esempi del nostro data set appartengano a 2 categorie GREEN e RED (i punti saranno così colorati):



Quello che può essere immediatamente notato dallo scatter plot è che i dati non possono essere separati andando a tracciare una retta sul grafo e questa osservazione pone l’attenzione su di una questione importante nel ML ovvero la distinzione tra problemi che sono linearmente separabili da quelli che non lo sono(come in questo caso).

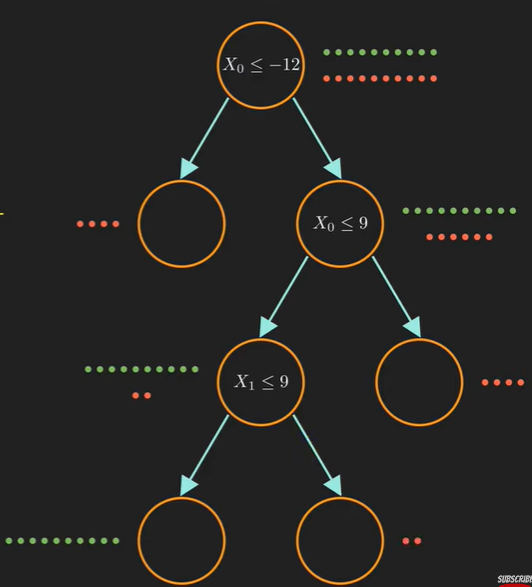
A Decision Tree (Albero di decisione) è un albero binario che va a dividere gli esempi del data set all’interno dei suoi nodi fino ad arrivare ad un nodo foglia il quale conterrà esempi appartenenti solo(o quasi solamente) ad una specifica classe di appartenenza.

Vediamo l’esempio grafico di un albero di decisione inerente ai dati precedentemente mostrati nello scatter plot:



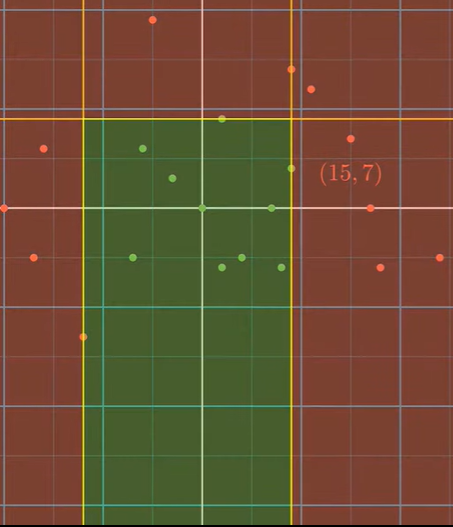
I nodi interni(ove presenti le condizioni di classificazione) risultano test da eseguire su delle features del campione dei dati, i nodi al momento vuoto rappresentano le fogli e in esse saranno raggruppati gli esempi a seconda dei test eseguiti.

In questo grafico vediamo il processo di smistamento effettuato da ogni nodo test lungo i percorsi che vanno dalla radice alle foglie:



E’ importante porgere l’attenzione nel processo di divisione per ogni test, infatti dopo il primo test quello contenuto nel nodo radice gli esempi del data set vengono smistati a sx nel nodo foglia che risulta contenere gli esempi la cui feature x0 risulta <= di -12 e come possiamo vedere tale foglia poiché contiene esempi appartenenti tutti alla stessa classe non necessiterà di un successivo passo di divisione per cui questi esempi sono già classificati come “rossi”, viceversa a dx i nodi rimanenti dovranno sottoporsi ad un successivo test poiché gli esempi restanti non appartengono tutti ad una stessa categoria. Il test node x0 <= 9 permetterà di dividere gli esempi tra quelli la cui feature x0 rispetta la condizione posti nel nodo dx ovvero un nodo foglia che anche in questo caso risulta raccogliere solo ed esclusivamente esempi di una classe la rossa e per tali ragioni risulta un nodo puro che non necessita di successive divisioni e resterà foglia poiché riesce a classificare questi esempi che ha raccolto. I restanti affronteranno il successivo test che sarà capace di classificare l’intera parte restante del data set.

L'albero di decisione appena mostrato risulta un modello di apprendimento che riesce a classificare i dati nelle due categorie ROSSI o VERDI e per farlo è come se avesse diviso lo spazio del problema visto con lo scatter plot in due aree un area alla quale tutti i futuri dati che apparterranno saranno classificati come verdi e un’altra area ove i dati che vi ricadranno saranno classificati come rossi:



Un albero di decisione può essere visto come un raggruppamento di if-else annidati, tuttavia questo sminuirebbe il concetto di Machine Learning dietro il funzionamento degli alberi di decisione, di fatti il problema principale sta in quali tipologie di test if-else eseguire e per capirlo occorre scomodare i concetti di Entropia dell’informazione e potere discriminante di una feature.

Infatti, non esiste un solo modo per poter classificare gli esempi del data set precedente attraverso un albero di decisione alcuni altri alberi possibili possono essere i seguenti:

Esempio: 1



Esempio: 2

Diagram, schematic

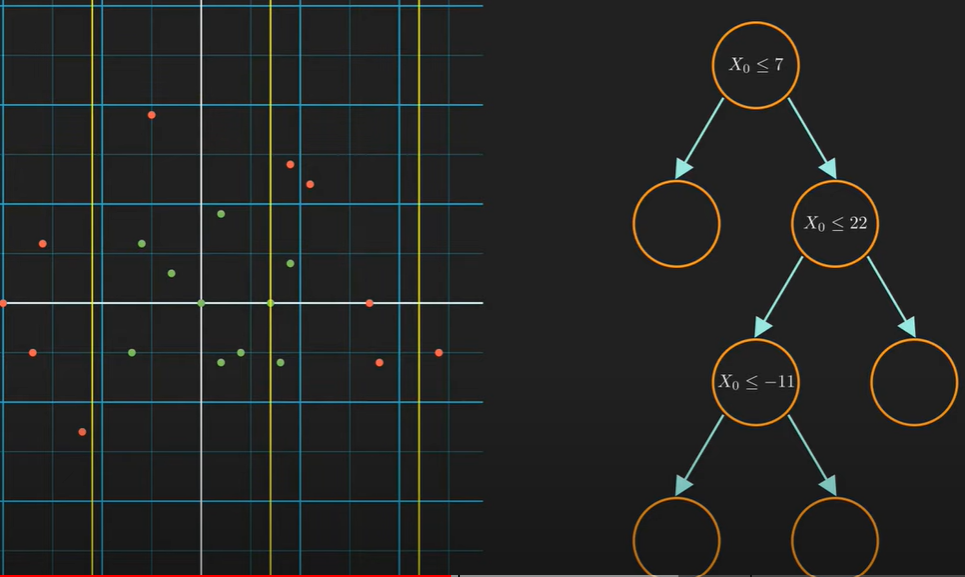
Description automatically generated

Esempio: 3

Diagram, schematic

Description automatically generated

Esempio: 4

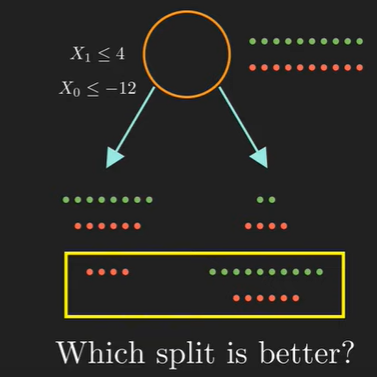


Esempio: 5



Il modello dell’albero decisionale ha necessità di apprendere quali sono le feature più importanti da prendere in considerazione per processare i dati e classificarli, e per farlo necessità ottimizzare questo processo di scelta al fine di ottenere una classificazione rapida (la quale è legata al numero di test fatti e quindi alla profondità dell’albero). Questo processo di ottimizzazione che apprende quali sono le migliori caratteristiche da sfruttare è il cuore del ML che si adopera con la scelta di un modello di apprendimento ad albero decisionale.

Ora supponiamo di andare a costruire il primo nodo dell’albero di decisione per l’esempio visto, quindi partiamo dal nodo di root che rappresenterà il primo test da eseguire sugli esempi per iniziarli a classificare. Per costruire tale nodo prendiamo in esame 2 test differenti tra di loro di modo che i risultati sia differenti in particolare con il primo test avremo diviso gli esempi iniziali in 2 sotto insieme i quali tuttavia risultano ancora eterogeni (contengono classi diverse) e con il secondo test invece riusciremo a dividere gli esempi in 2 insiemi di cui questa volta uno avrà esempi appartenenti ad una sola classe, mentre l’altro risulterà eterogeneo ora sorge normale la domanda quale dei 2 test scegliere?



Per trovare la risposta dovremmo ricorrere alla teoria dell’informazione e al concetto di entropia.

Il modello, infatti, per scegliere il test migliore calcolerà L’INFORMATION GAIN e quello che avrà un valore di tale misura maggiore sarò preferito ad un altro test.

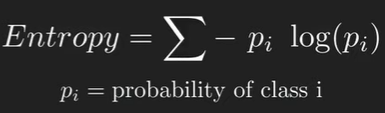
Icon

Description automatically generated

Il nostro obiettivo è riuscire a dividere in due classi il dataset che vediamo sopra al momento esso poiché ha un equale numero di esempi verdi e rossi ha il massimo valore di Impurità (misurata come Indice di Gini o impurità di Gini, in onore dello statista Italiano Corrado Gini) il nostro obiettivo di classificazione coincide nel abbassare (o addirittura azzerare) l’indice di impurezza di Gini.

Con la massima impurità che abbiamo al momento se provassi a predire un esempio a quale classe appartiene avremo 0.5 (50%) di possibilità di classificare correttamente.

Se L’entropia (che è una misura del disordine è alta) allora siamo abbastanza incerti a quale classe l’esempio appartenga, viceversa se l’entropia è bassa (allora c’è poco disordine) e avremo le idee più chiare su quale sia la classe da assegnare. Possiamo pensare all’entropia come disordina, ma cosa significa? Collegandoci all’esempio visto il disordine è comparato ad avere un insieme eterogeneo di esempi più l’insieme è eterogeneo è più sarà disordinato ovvero sarà alta la sua entropia, meno un sistema sarà eterogeno e quindi più si avvicinerà ad essere un insieme omogeneo e più basso sarà il disordine ovvero l’entropia sarà bassa.



Stando a questa formula e andando a calcolare l’entropia dell’insieme di partenza con esempi in equal numero delle 2 classi e quindi una probabilità per un esempio scelto random di appartenere alla classe i di ½ = 0.5 avremo che la formula dell’entropia:

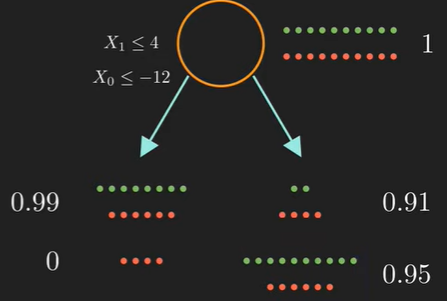
A picture containing text, device, gauge, meter

Description automatically generated = 1

Chart, line chart

Description automatically generated

Il risultato sarà 1 massimo valore che può essere raggiunto dall’entropia. Il logaritmo usato è di base 2, ora andando a calcolare l’entropia sugli insiemi dati dopo le nostre condizioni fissate avremo che:



E come è possibile vedere dall’immagine usando il test x0 <= -12 si riesce ad avere un insieme la cui entropia è 0 ed è per questo che viene detto Pure Node(nodo puro perché non presenta disordine gli elementi sono tutti omogenei).

Ora una volta compreso il concetto di entropia dobbiamo usarlo per ricavare la misura di INFORMATION GAIN calcolata come l’entropia del nodo genitore(prima del test) – la somma delle entropie dei figli di tale nodo:



Nella formala soprastante vediamo che l’entropia viene ad essere moltiplicata per un peso e questo peso è dato dalla divisione degli esempi ovvero il rapporto tra gli esempi presenti nel nodo figlio in questione fratto gli esempi totali presenti nel nodo padre (quindi risulta essere il rapporto tra le dimensioni del figlio fratto quelle del padre). Andiamo ad effettuare i calcoli per le 2 scelte di test:

**ES: test 🡪 x0 <= 4**

A picture containing text, device, gauge

Description automatically generated

Stando alla formula calcoliamo l’entropia del padre come visto in precedenza ed è 1 e da questa sottraiamo la sommatoria dei pesi moltiplicati per le rispettive entropie dei nodi figlie ovvero:

figlio sx: peso = N(esempi nel nodo figlio) / N(esempi nel padre) = 14/20

entropia del figlio sx: meno probabilità di appartenere alla classe verde per il logaritmo di base 2 di tale probabilità + meno la probabilità di appartenere alla classe rossa per il logaritmo della classe rossa:

-(8/14)\*log2(8/14) + -(6/14)\* log2(6/14) =

- 0.571428 \* (- 0,46) + - 0.428571428 \* (- 0,52) =

0,69 + 0.23 = 0,99

Da cui avremo wfiglioSX \* Eentropia fsx = 14/20 \* 0,99

Effettuando lo stesso processo ricaveremo

IG = 0, 034

**ES: test 🡪 x0 <= -12**



Svolgendo gli stessi calcoli otterremo una Information gain IG = 0,24

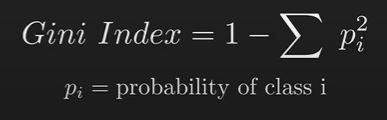
Text

Description automatically generated with low confidence

Scegliamo quindi lo split che permette di avere un maggiore valore di information gain. L’esempio visto ci ha permesso di capire come il modello sceglie di costruire l’albero migliore per farlo dovrà calcolare l’information gain tra tutti i possibili Split test e scegliere quelli con maggiore IG. Nel fare questa scelta di ottimizzazione l’albero di decisione adotta un algoritmo di ricerca greedy, ne consegue che i successivi split dipenderanno dal precedente e questo potrebbe essere un problema poiché gli algoritmi greedy hanno la vista corta (ovvero fanno la migliore scelta sul momento attuale e non nel lungo tempo, questo significa che ci potrebbero essere delle combinazioni di split che partendo da uno a minore IG potrebbero portare alla formazione di un albero di classificazione più efficiente in termini di test e tempo).

Ad ogni livello dell’albero l’impurità presenti in ogni nodo diminuisce.

Vediamo adesso l’indice di Gini:



Esso è calcolato come 1 meno la sommatoria del quadrato delle probabilità di appartenere ad ogni classe [l’indice di Gini è utilizzato per efficientare il calcolo poiché nella forma del IG compare il logaritmo che è più complesso da calcolare rispetto al semplice quadrato della formula di Gini].

Riprendendo l’esempio precedente andiamo a calcolare l’indice di Gini ad ogni nodo test:

Nodo root comune =0,5

**ES: test 🡪 x0 <= 4**

Figliosx(8v+6r) Text

Description automatically generated=0,49

Figliodx(2v +4r) Text

Description automatically generated=0,44

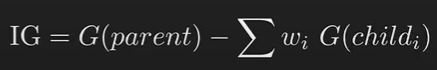
**ES: test 🡪 x0 <= -12**

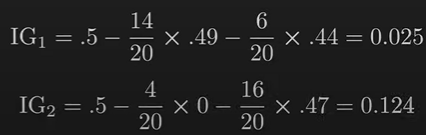
Figliosx(4r)=0

Figliodx(10v +6r) Text

Description automatically generated=0,47

Una volta ottenuti ottenuti gli indici di Gini per ogni nodo e per ogni tipologia di test utilizziamo l’indice di Gini al posto della formula dell’entropia la quale contiene il logaritmo e sostituimo nella forma dell’Information Gain ottenendo:





Icon

Description automatically generated with medium confidence

Otteniamo anche in questo caso lo stesso risultano rispetto all’uso dell’entropia.

**REGRESSION TREE REGRESSOR**

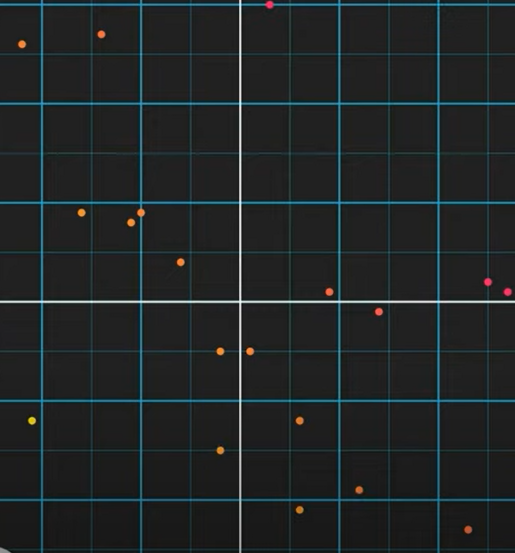
Partiamo da 2 features x0 e x1 e il target y che rappresentano le caratteristiche del nostro dataset:

****

Per comodità andiamo a rappresentare le features su di un piano cartesiano bidimensionale con le due features x0 e x1, mentre il target sarà rappresentato dal colore:

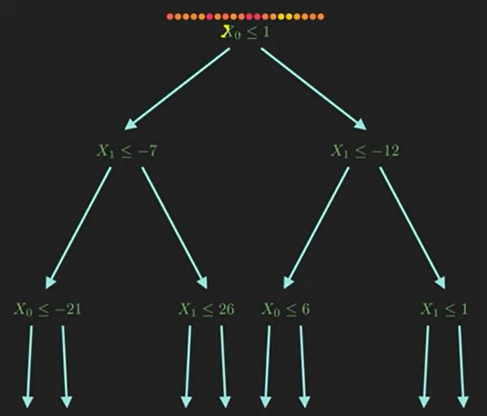
Rectangle

Description automatically generated with medium confidence

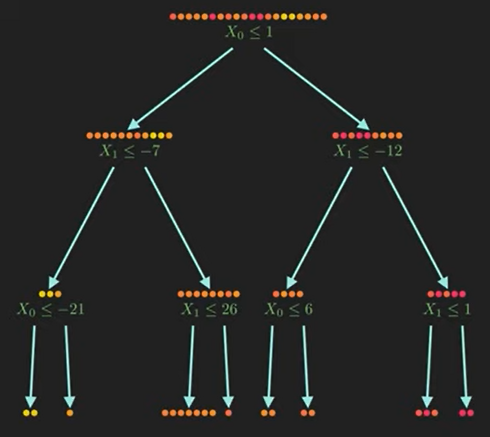


Come ben sappiamo un problema di regressione consiste nel trovare la retta di best fit con i dati la quale permette di tracciare una retta che minimizza la distanza da tutti i punti del piano. Il nostro obiettivo è quello di risolvere il problema di regressione usando un albero di decisone binario.

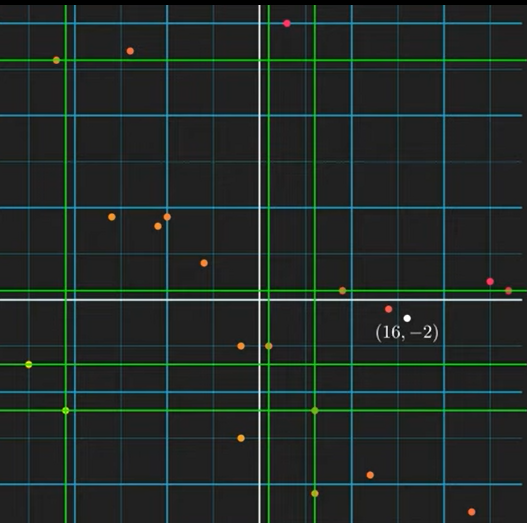
Vediamo come costruire l’albero binario di regressione e come esso andrà a risolvere il problema:



All’inizio come è possibile vedere dalla figura tutti i nostri dati ovvero i punti con diverse gradazioni di colore sono posizionati nel primo nodo pronti per essere sottoposti al primo test presente nel nodo root, successivamente al primo test gli esempi verranno suddivisi in base alla condizione e ricorsivamente rispetto alle successive ricorsione sarà ripetuto lo stesso procedimento.



Questo è il risultato che otteniamo dopo il processo.



Come possiamo vedere lo spazio del problema è stato diviso più volte da linee separatrice (di colore verde in figura) queste linee stanno ad indicare il test fatto da ogni nodo che permette di partizionare l’insieme di partenza in 2 sotto insiemi la ripetizione di più test va a creare sottospazi del problema iniziale.

Ora supponiamo di avere un nuovo esempio (x0=16, x1=-2) questo elaborato dal albero finirà nel figlio sx foglia di x1<1, ora un problema di regressione si risolve fornendo una predizione che rappresenti un numero reale e per farlo in tal caso useremo gli esempi presenti in questo nodo foglia ed effettueremo su di loro un calcolo di media il quale risulterà essere il valore di predizione.

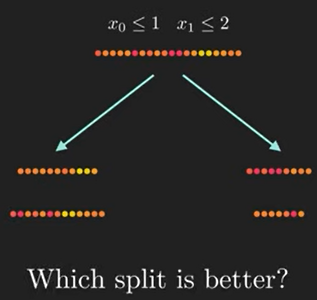
A black screen with white text

Description automatically generated with low confidence

Anche in questo caso si finirà con uno spazio del problema suddiviso in regione alle quali sarà affidato un colore il colore che sarà affidato in questo caso rappresenterà la stessa gradazione che abbiamo data al valore target di fatto la colorazione di un area dello spazio sarà mappata ad un preciso valore della gradazione del colore della feature target .



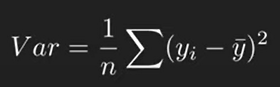
Adesso dobbiamo capire anche in questo caso come scegliere al meglio i test da effettuare per la regressione:



Nell’esempio è possibile vedere due test a confronto il 1°: x0<=1 i cui risultati sono rappresentati dalla prima riga di punti nei nodi figli e il 2°: x1<=2 ove i risultati sono riportati nella seconda ed ultima riga nei nodi figli. Per effettuare una giusta scelta dobbiamo capire quale dei 2 test va a definire uno split che vada a ridurre l’impurità dell’insieme degli esempi presenti nei figli e per capirlo ci serve la VARIANCE REDACTION ovvero il fattore di riduzione della varianza del campione.

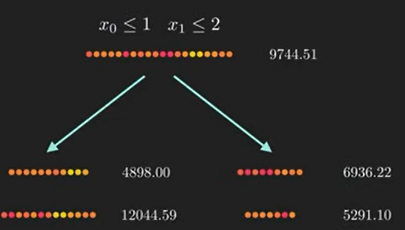
Nei problemi di regressione con gli alberi di decisione usiamo la VARIANZA come misura dell’impurità degli esempi nel campione, in un modo simile a quello che abbiamo fatto con l’entropia e l’indice di Gini per il problema di classificazione.

**Calcolo della varianza:**



La formula è calcolata rispetto ad ogni nodo come n è il numero degli esempi presenti nel nodo ove calcoliamo la varianza yi risulta essere uno dei target degli esempi presenti nel nodo mentre  risulta la media dei target degli esempi nel nodo e la sommatoria e fatta sugli esempi appartenenti al nodo preso in considerazione.

I valori di varianza per ogni test nei nodi dell’albero sono:

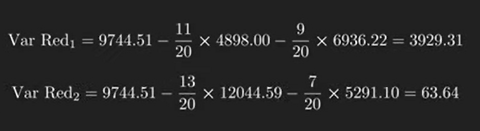


Una volta ottenute le varianze andiamo finalmente a calcolare la misura che ci permetterà di effettuare una scelta sui test, ovvero calcoliamo la VARIANCE REDACTION come:

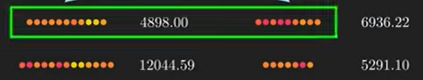


La formula è calcolata come la varianza presente nel nodo padre – la somma delle varianze presenti nei nodi figli.

I risultati per i 2 test risultano essere:







Per cui il primo test di split è da preferire al secondo. Ora quale che sia il migliore tra tutti i possibili split è di nuovo come nei problemi di classificazione un problema di ricerca che in particolare il modello di apprendimento supervisionato ad albero di decisione va ad affrontare con una ricerca locale di tipo greedy.

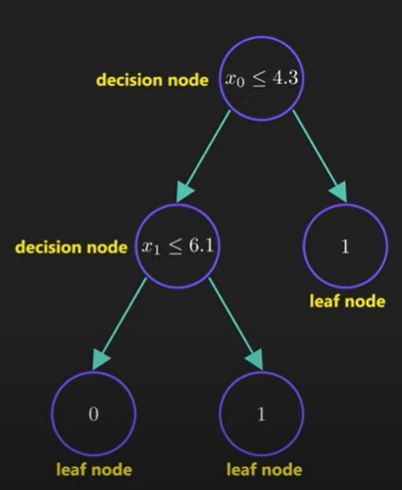
**RANDOM FOREST ALGORITHM**

A

Il seguente data set ha 6 features e la feature target la y ha range binario 0 o 1, quindi siamo difronte ad un problema di classificazione.



Se usassimo un semplice albero di decisione come visto negli esempi precedenti esso risulterebbe come il seguente:



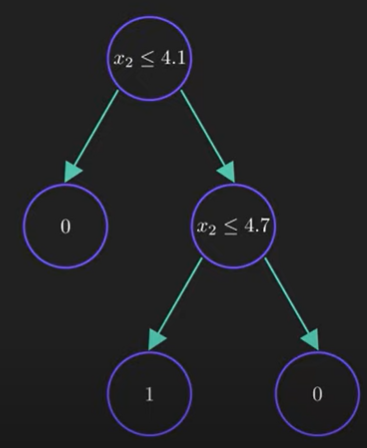
Ora cerchiamo di capire perché il decision tree potrebbe non andare bene in ‘articolare prendiamo dal precedente data set l’esempio con id=1:



E andiamo a modificare le feature x0 e x1 ottenendo così:



Adesso andiamo ad addestrare l’albero di decisione su questo data set modificato e otterremo un albero di decisione differente dal precedente e questo ci fa capire che il modello di apprendimento ad albero decisionale è strettamente legato ai dati di training con i quali viene addestrato (in particolare l’albero di decisione soffre del problema noto come overfitting dei dati). Per questo esempio con le modifiche apportate avremo il seguente albero:



E’ facile notare la differenza strutturale tra i 2 alberi che potrebbero essere di molto più marcata per alberi decisionali con maggiore profondità. Questo fa si che il modello non riesca a generalizzare correttamente per poi comportarsi nel modo voluto e desiderato durante il processo di classificazione per tali ragioni per cercare di risolvere questo problema si introducono le RANDOM FOREST(foreste casuali) ovvero un assemblamento di alberi decisionali strutturalmente differenti tra di loro ma addestrati su di uno stesso data training.

Per ottenere una foresta casuale si parte prima dal maneggiare il dataset di partenza in particola si sceglie un numero casuale di esempi presi dal dataset in modo casuale per formare un nuovo dataset che sarà dato come training set ad uno degli alberi della foresta, questo aumenta la varianza degli alberi e il principio sta nel mediare alla fine i risultati ottenuti da tutti gli alberi della foresta seguendo quello che è conosciuto come principio “ del wisdom of th crowd”.

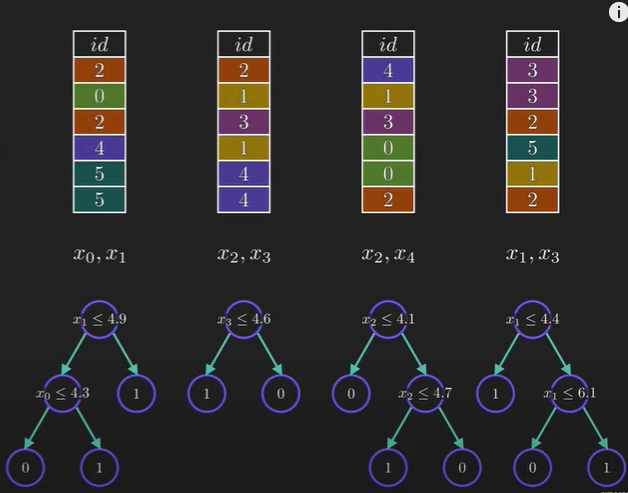
Il processo descritto per ottenere i vari training set per ogni albero è detto Bootstrapped Dataset( o semplicemente bootstrap) anche chiamato bagging:

Graphical user interface, application

Description automatically generated

Se il processo utilizza la casualità per selezionare gli esempi per creare un training set da dare ad un albero della foresta, ma non permette il reinserimento di esempi già scelte allora il processo prende il nome di PASTING.

Inoltre, prenderemo per ogni training set che daremo ad un albero solo alcune features e altre no in modo da aumentare la diversità degli alberi.



Questi risultano essere i diversi alberi generati dal processo e assieme costituiscono una RANDOM FOREST. Ora per effettuare la predizione su di un nuovo esempio, esso l’esempio sarà dato in pasto ad ogni albero della foresta il quale produrrà una predizione in base alle features sulle quali è stato addestrato quando tutti gli alberi della foresta avranno visto lo stesso esempio si dovranno aggregare le predizioni fatte tramite un processo di AGGREGAZIONE dei risultati per tirare fuori la predizione finale, il processo di aggregazione può essere una semplice media, la moda, la media oppure una media pesata in base all’importanza delle features sulle quali l’albero è stato costruito. Per i problemi di classificazione si usa la maggioranza degli esempi classificati su di una classe.

Questa strategia di costruzione della foresta tramite il processo di Bootstrapping permette di avere casualità negli esempi scelti e quindi una diversità che porterà a varianza, ma non solo la seconda scelta casuale quella che prevede l’uso di differenti featureas per la costruzione dei singoli alberi della foresta porterà a donare agli alberi indipendenza e questi due aspetti combinati riusciranno ad affrontare al meglio il problema dell’overfitting.

Qual è il numero di features ideale per effettuare il bootstrapping? Ricerche hanno dimostrato che un buon inizio per la scelta del numero potrebbe essere la radice quadrata della totalità delle features oppure il logaritmo della totalità delle features, ad ogni modo resta una decisione che per essere presa necessita di una scelta empirica condotta sui risultati ottenuti in base alla decisione fatta.