**CLASSIFICATORE: KNearestNeighbour Custom**

Come classificatore semplice custom abbiamo deciso di utilizzare il classificatore KNearestNeighbour.

Il **KNN** fa parte dei classificatori **Instance-based** (noti anche come Lazy Learners) i quali sfruttano un apprendimento basato sulla memorizzazione dei dati di addestramento forniti, utilizzandoli come riferimento e confronto per la classificazione di nuove istanze.

Questo confronto utilizza una particolare **misura di prossimità**: usano la distanza o similarità tra i vari esempi.

Per creare e utilizzare un modello come il KNN per prima cosa è necessario memorizzare al suo interno l’intero training set (questa in pratica è la sua fase di addestramento) dopodichè per ciascun nuovo record da classificare viene calcolata la sua **distanza** da ciascun oggetto del training set e infine si individuano i k record più vicini: al nuovo record è assegnata la classe prevalente tra questi ultimi.

Abbiamo deciso di utilizzare questo classificatore perché:

* Non richiede l’induzione di un modello, con conseguente risparmio di risorse e **semplicità**.
* Non fa ipotesi sui dati (visto che si basa solo sulla misura di prossimità), di conseguenza è possibile utilizzarlo in un’ampia gamma di problemi, anche quando il dataset non è divisibile in maniera lineare(quindi è molto **flessibile**)
* Su dataset piccoli può comunque mantenere una certa accuratezza, che altri algoritmi invece fanno fatica a mantenere.

Ovviamente teniamo in considerazione anche i suoi lati negativi ovvero:

* Ogni volta occorre calcolare la prossimità tra il nuovo record e gli esempi del training set e ciò comporta che richieda più risorse computazionali rispetto ad altri modelli.
* Richiede molta memoria per memorizzare tutti i dati all’interno.
* Potrebbe ignorare o considerare meno attributi con una scala di valori inferiore di conseguenza è necessario quindi una normalizzazione o standardizzazione

Il punto cruciale quando si vuole utilizzare un classificatore di questo tipo è quindi effettuare un adeguata normalizzazione degli attributi e individuare il valore di k e la misura di prossimità più adatti.

Nella pratica tutto ciò è stato implementato in codice python senza l’utilizzo di alcuna funzione esterna.

La funzione **fit()** permette di immagazzinare i dati nel modello creato mentre la funzione **predict()** permette di effettuare le predizioni sui dati in ingresso. Infine si è implementata la funzione **calcolo\_distanza()** che permette di effettuare il calcolo della ‘distanza’ tra i record.

**DETERMINAZIONE VALORI OTTIMALI DI K e MISURA PROSSIMITA’**

Il tuning di questi iperparametri ha tenuto conto di questi possibili valori:

* K valutato in un range da 1 a 20
* Misure prossimità: ‘distanza euclidea’ e ‘distanza di manhattan’

Per effettuare l’ottimizzazione si è eseguito il modello per ogni possibile combinazione di questi iperparametri al fine di individuare quella che restituisce la migliore performance.

Il test è stato effettuato addestrando il modello con il training set e verificando le performance su un validation set ( entrambi ottenuti da uno split sul training set ottenuto inizialmente).

**ESECUZIONE DEL KNN CUSTOM**:

Una volta individuato il miglior valore per k e la migliore misura di prossimità tra quella variate, si è proceduto all’effettiva esecuzione del classificatore addestrandolo con l’interno training set e applicandolo al test set.

**RISULTATI**:

Immagine che contiene testo, schermata, Carattere, documento

Il contenuto generato dall'IA potrebbe non essere corretto.

**ANALISI DEI RISULTATI:**

Dai risultati si ottiene che il miglior numero di elementi vicini da considerare per la predizione è 3 e che la misura di prossimità ideale sia quella di manhattan.

Il modello è addestrato su **dati non normalizzati** quindi sicuramente in seguito alla normalizzazione i risultati saranno migliori (infatti abbiamo la presenza di attributi con scale di valori molto diverse).

Un altro problema evidente è lo **sbilanciamento del dataset** che comporta appunto classi con pochi record. Il nostro classificatore, valutando i 3 elementi più vicini, potrebbe aver assegnato il record alla classe errata data proprio questa disparità di elementi. Comunque, avendo un k non eccessivamente grande, dovremmo aver perlomeno diminuito tale possibilità. Sarà comunque necessario quindi un bilanciamento del dataset.

Si nota poi che South America e North America non vengono mai predette correttamente, e ciò risulta molto interessante se si considera che insieme contengono una buona parte dell’intero training set. Si suppone quindi che le caratteristiche utilizzate potrebbero non essere sufficientemente distintive e perciò il classificatore non riesce a dividere i record adeguatamente. Vedremo in seguito se con il preprocessing le caratteristiche potranno essere più ottimali per la corretta separazione delle classi.

Osservando invece l’Europa o l’Africa si può notare che precision si aggira intorno al 50% , che significa che circa il 50% dei record predetti con quelle classi si è rivelato errato, sarà quindi necessario come detto in precedenza cerca di differenziare meglio le classi.