Data and Web Mining

Raccolta domande teoriche

Quali sono le differenze tra gli algoritmi di supervised learning e quelli di unsupervised learning?

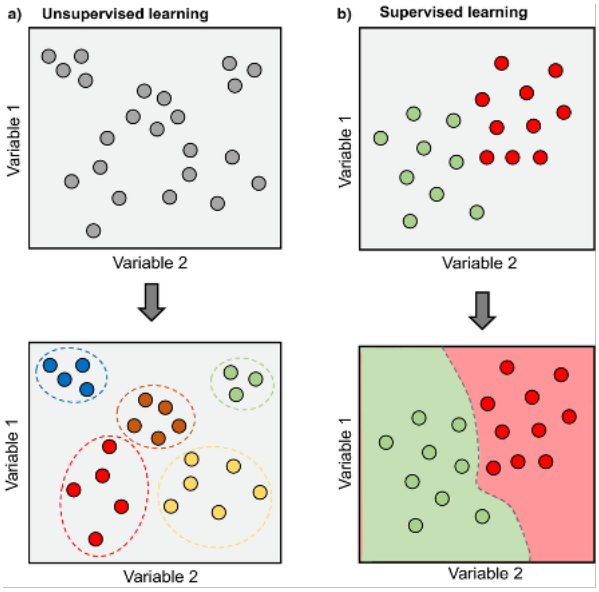
Supervised Learning:

- Obiettivo: Predire un'etichetta di output basata su una o più caratteristiche di input.
- Esempi di Applicazioni: Diagnosi mediche, riconoscimento vocale, riconoscimento di immagini, previsione del prezzo delle azioni.
- Valutazione del Modello: È possibile valutare la precisione del modello confrontando le previsioni del modello con le etichette vere.
- Complessità Computazionale: Tendenzialmente più elevata rispetto all'apprendimento non supervisionato, a causa della necessità di addestrare il modello con un grande numero di esempi etichettati.
- Overfitting: È un rischio maggiore in quanto il modello potrebbe adattarsi troppo ai dati di addestramento e perdere la capacità di generalizzare su dati nuovi e non visti.
- Esempio algoritmi: Decision Trees, Support Vector Machines, Neural Networks, Linear Regression.

Unsupervised Learning:

- Obiettivo: Esplorare la struttura sottostante o le relazioni tra le variabili nei dati.
- Esempi di Applicazioni: Segmentazione del mercato, organizzazione di grandi biblioteche di documenti, compressione di immagini.
- Valutazione del Modello: È più difficile valutare l'efficacia del modello, poiché non ci sono etichette vere con cui confrontare le previsioni o i raggruppamenti del modello.
- Complessità Computazionale: Tendenzialmente meno elevata rispetto all'apprendimento supervisionato, ma può variare a seconda dell'algoritmo e del numero di dati.
- Scoperta di Conoscenza: È particolarmente utile quando non si conoscono le etichette o quando si desidera scoprire relazioni non note tra i dati.
- Esempio algoritmi: K-Means, Hierarchical Clustering, DBSCAN, t-SNE.

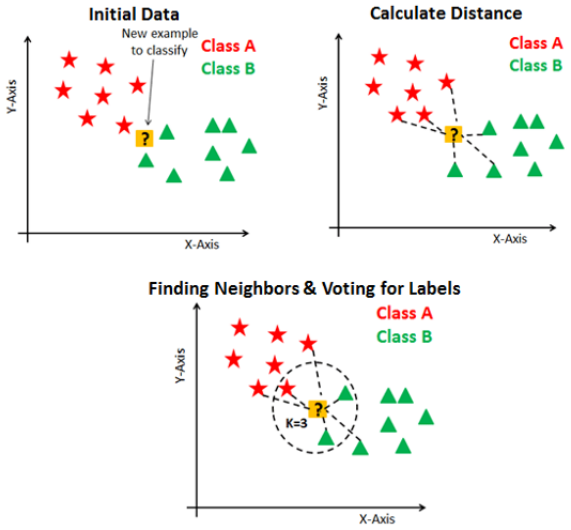
In generale, gli algoritmi di apprendimento supervisionato sono utilizzati per problemi di classificazione e regressione, mentre gli algoritmi di apprendimento non supervisionato sono utilizzati per problemi di clustering e riduzione della dimensionalità.



Descrivere l'algoritmo K-nn e spiegare cosa succede cambiando il parametro k

L'algoritmo K-nn, o K-Nearest Neighbors, è un metodo di apprendimento supervisionato utilizzato sia per la classificazione che per la regressione. Il suo funzionamento è intuitivo: dato un nuovo punto da classificare o da cui prevedere un valore, l'algoritmo identifica i k punti più vicini nel dataset di addestramento e assegna la classe o il valore medio di questi vicini al nuovo punto.
Funzionamento:

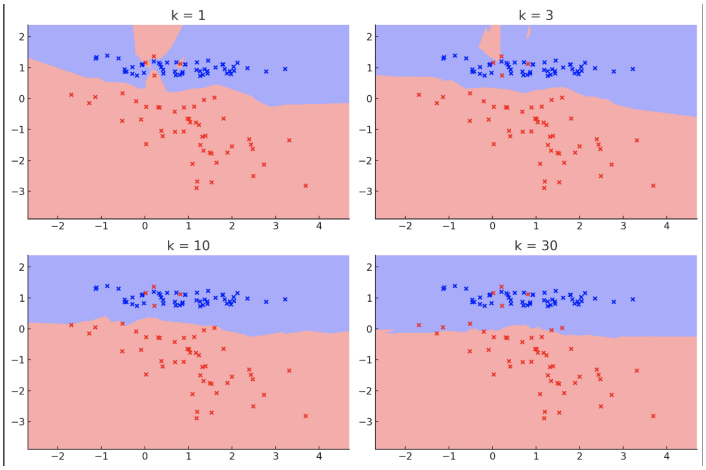
- Initial Data: Si parte con un dataset di addestramento dove ogni punto è etichettato con una classe o un valore.
- Calculate Distance: Per ogni nuovo punto, si calcola la distanza da tutti i punti nel dataset di addestramento. La distanza euclidea è comunemente utilizzata, ma altre metriche di distanza possono essere applicate a seconda del contesto.
- Finding Neighborhood & Voting for Labels: Si selezionano i k punti più vicini e, in base al task:
  - Classificazione: Si assegna la classe più frequente tra i k vicini (majority voting).
  - Regressione: Si assegna la media dei valori dei k vicini.



Il valore di k è cruciale. Un k troppo piccolo rende l'algoritmo sensibile al rumore, mentre un k troppo grande lo rende insensibile alle variazioni locali. È comune utilizzare un numero dispari per k in task di classificazione per evitare pareggi nel voting. Inoltre, è possibile attribuire pesi diversi ai vicini in base alla loro distanza dal nuovo punto, dando più importanza ai punti più vicini.

Per migliorare le prestazioni, è consigliabile scalare le feature in modo che abbiano tutte lo stesso range di variazione, utilizzando tecniche come MinMax Scaler o StandardScaler. Questo perché le feature con variazioni più ampie potrebbero dominare quelle con variazioni più ridotte nel calcolo delle distanze.

L'algoritmo K-nn è semplice ed efficace, soprattutto con dati numerici e quando si dispone di una metrica di distanza appropriata. Tuttavia, ha un costo computazionale elevato, soprattutto con dataset di grandi dimensioni, poiché richiede il calcolo della distanza da tutti i punti del dataset per ogni nuovo punto.



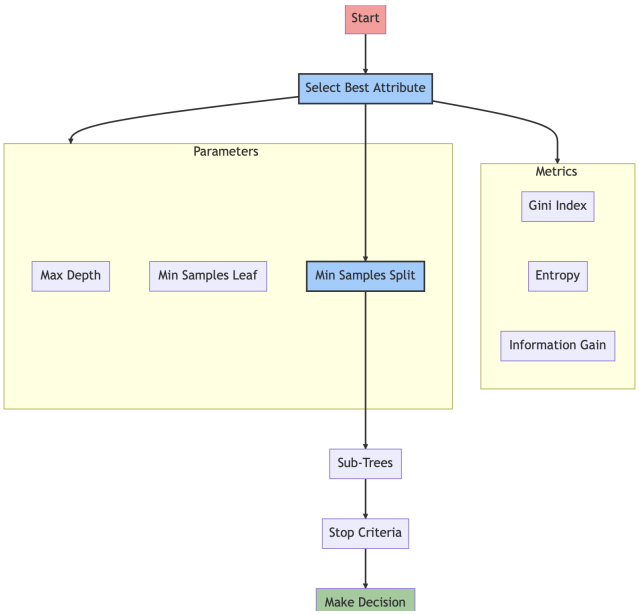
L'immagine mostra come l'algoritmo K-nn varia con diversi valori di k, illustrando l'effetto di overfitting con k piccoli e di underfitting con k grandi. Nello specifico abbiamo ogni grafico che rappresenta un diverso valore di k e mostra come l'algoritmo classifica i punti in due classi diverse (colori diversi nello sfondo) basandosi sui punti del dataset (punti colorati nel grafico).

- Nel primo grafico, con $k = 1$, si nota un evidente overfitting, con una frontiera di decisione molto irregolare che segue strettamente i punti del dataset.
- Nel secondo grafico, con $k = 3$, la frontiera di decisione è leggermente più liscia ma ancora abbastanza irregolare.

- Nel terzo grafico, con $k = 10$, la frontiera di decisione è più liscia e generalizzata, ma potrebbe ancora catturare bene la struttura dei dati.
- Nel quarto grafico, con $k = 30$ la frontiera di decisione è molto semplice e liscia, il che potrebbe indicare un potenziale underfitting, dove il modello non cattura adeguatamente la struttura sottostante dei dati.

Descrivere l'algoritmo per la costruzione di un Decision Tree e spiegare le metriche e i vari parametri

Gli Alberi di Decisione sono modelli di apprendimento supervisionato utilizzati per problemi di classificazione e regressione. Un albero di decisione divide ricorsivamente lo spazio degli input in regioni omogenee, per poi assegnare una classe (o un valore, in caso di regressione) alla regione in cui cade un nuovo input.



Algoritmo di Costruzione: la costruzione di un albero di decisione inizia con la radice, che contiene l'intero dataset di addestramento. Il dataset viene poi diviso in sottoinsiemi omogenei basandosi su un attributo e un valore di soglia. Questo processo si ripete ricorsivamente su ogni sottoinsieme, creando nuovi nodi, fino a quando tutti i dati in un nodo appartengono alla stessa classe o fino a quando non sono soddisfatti altri criteri di arresto.

Criteri di Divisione: i criteri di divisione, come l'Entropia e l'Indice Gini, misurano l'impurità di un nodo. Un nodo è puro quando tutti i suoi dati appartengono alla stessa classe. Il miglior attributo e valore di soglia da utilizzare per dividere un nodo sono quelli che riducono al massimo l'impurità.

- **Entropia:** Misura dell'incertezza o del disordine in un nodo.

$$\text{Entropia}(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

- **Guadagno di Informazione:** Differenza tra l'entropia del nodo padre e la somma ponderata delle entropie dei nodi figli.

$$\text{Guadagno}(S, A) = \text{Entropia}(S) - \sum_{v \in \text{Val}(A)} \frac{|S_v|}{|S|} \text{Entropia}(S_v)$$

Criteri di Arresto: alcuni dei criteri di arresto includono la profondità massima dell'albero, il numero minimo di campioni per nodo e il numero minimo di campioni per foglia.

Metriche di Valutazione: le metriche di valutazione, come l'Accuracy, la Precision, la Recall e l'F1 Score, aiutano a quantificare le prestazioni di un modello di classificazione.

- **Accuracy:** Rapporto tra le previsioni corrette e il totale delle previsioni.
- **Precision:** Rapporto tra i veri positivi e la somma dei veri positivi e dei falsi positivi.
- **Recall:** Rapporto tra i veri positivi e la somma dei veri positivi e dei falsi negativi.
- **F1 Score:** Media armonica tra precision e recall.

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Parametri del Modello: alcuni parametri chiave che influenzano la costruzione dell'albero di decisione sono la profondità massima dell'albero, il numero minimo di campioni richiesti per dividere un nodo interno e il numero minimo di campioni richiesti per essere presenti in un nodo foglia.

Cosa cambia in un decision Tree utilizzato in un task di classificazione a un decision Tree utilizzato per un task di regressione?

Un *Decision Tree* è un modello di apprendimento supervisionato utilizzato sia per problemi di classificazione che di regressione. La principale differenza tra i due tipi di alberi risiede nella natura della variabile obiettivo e nelle metriche di errore utilizzate per effettuare le divisioni.

Decision Tree per Classificazione
Nel contesto della classificazione, diverse metriche possono essere utilizzate per valutare la qualità delle divisioni:

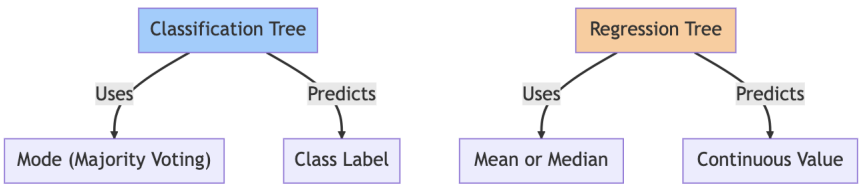
- **Classification Error:**
$$\text{Error} = 1 - \max(p_1, p_2, \ldots, p_m)$$
dove p_i rappresenta la proporzione di campioni appartenenti alla classe i in un nodo.
- **Information Gain:**
$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Val}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$
dove l'entropia è definita come:
$$\text{Entropy}(S) = -\sum_{i=1}^{m} p_i \log_2 p_i$$
- **Gain Ratio:**
$$\text{GainRatio}(S, A) = \frac{\text{Gain}(S, A)}{\text{SplitInformation}(S, A)}$$
con:
$$\text{SplitInformation}(S, A) = -\sum_{v \in \text{Val}(A)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$$
- **Gini Index:**
$$\text{Gini}(S) = 1 - \sum_{i=1}^{m} p_i^2$$
dove p_i è la proporzione di campioni appartenenti alla classe i in un nodo.

Decision Tree per Regressione
Nel contesto della regressione, l'obiettivo è prevedere un valore continuo piuttosto che una classe. La metrica di errore comunemente utilizzata per la divisione è il *Mean Squared Error* (MSE):

$$\text{MSE}(S) = \frac{1}{|S|} \sum_{i \in S} (y_i - \bar{y})^2$$

dove y_i è il valore obiettivo del campione i e $\bar{y}$ è la media dei valori obiettivo in S.

La principale differenza nella struttura dell'albero tra classificazione e regressione risiede nelle foglie: in un albero di regressione, una foglia contiene la media dei valori obiettivo dei campioni in essa, mentre in un albero di classificazione, contiene la classe maggioritaria.



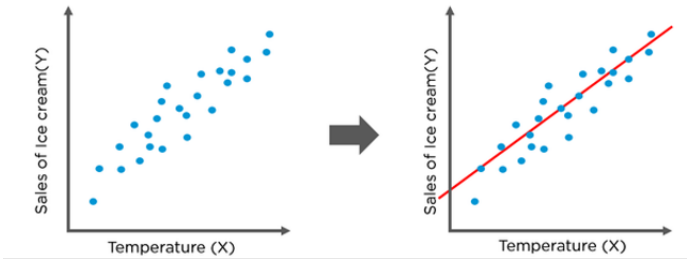
Descrivere un algoritmo efficiente per costruire un decision Tree

Un algoritmo efficiente per costruire un Decision Tree può essere realizzato utilizzando un approccio iterativo e una coda di priorità. L'algoritmo proposto si basa sulla massimizzazione del gain informativo a ogni passo per decidere il miglior attributo su cui effettuare lo split. Ecco i passi fondamentali dell'algoritmo:

```
1. Inizializza albero con nodo radice contenente tutto il training set
   Inizializza coda di priorità PQ

2. Per ogni nodo N nell'albero:
       Per ogni possibile split S di N:
           Calcola gain informativo di S
           Inserisci (N, S, gain) in PQ

3. Finché PQ non è vuota:
       Estrai (N, S, gain) da PQ con il massimo gain
       Esegui split S su nodo N
       Crea nodi figli destro e sinistro

4. Ripeti i passi 2 e 3 per ogni nuovo nodo creato
   Finché non sono soddisfatti i criteri di arresto

5. Se PQ è vuota o sono raggiunti i criteri di arresto:
       Termina algoritmo
```

Questo approccio consente una costruzione efficiente e ottimizzata dell'albero, garantendo che, ad ogni passo, si scelga lo split che massimizza il gain informativo. Tuttavia, è importante considerare gli iper-parametri e i criteri di arresto, in quanto possono influenzare significativamente la struttura dell'albero finale, potenzialmente portando a overfitting o underfitting del modello rispetto ai dati di addestramento.

Descrivere la regressione lineare e le metriche



La *regressione lineare* è un modello statistico ampiamente utilizzato, particolarmente adatto per task di regressione. Il modello è apprezzato per la sua semplicità, interpretabilità e adattabilità ai dati.
Questo approccio mira a trovare la retta, o in generale un iperpiano, che meglio approssima la distribuzione dei dati nel piano cartesiano. La forma della retta in due dimensioni è data da:

$$y = mx + b$$

dove:

- y è la variabile dipendente (risposta),
- x è la variabile indipendente (predittore),
- m è il coefficiente angolare (pendenza della retta),
- b è l'intercetta (punto in cui la retta interseca l'asse y).

L'obiettivo della regressione lineare è trovare i valori di m e b che minimizzano l'errore quadratico medio (MSE) tra i valori osservati e quelli predetti da modello:

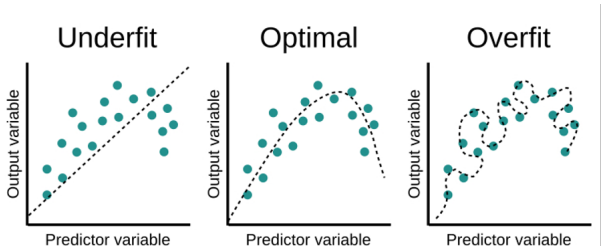
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

In alcuni casi, un modello lineare potrebbe non essere sufficientemente espressivo per catturare la complessità dei dati. In tali situazioni, si può ricorrere alla regressione polinomiale, che modella la relazione tra la variabile indipendente x e la variabile dipendente y come un polinomio di grado d:

$$y = a_d x^d + a_{d-1} x^{d-1} + \cdots + a_1 x + a_0$$

Tuttavia, aumentare eccessivamente il grado del polinomio può portare a overfitting, soprattutto quando il grado del polinomio è uguale al numero di punti nel dataset, rendendo il modello troppo complesso e incapace di generalizzare bene su dati non visti.

Cos'è l'overfitting?



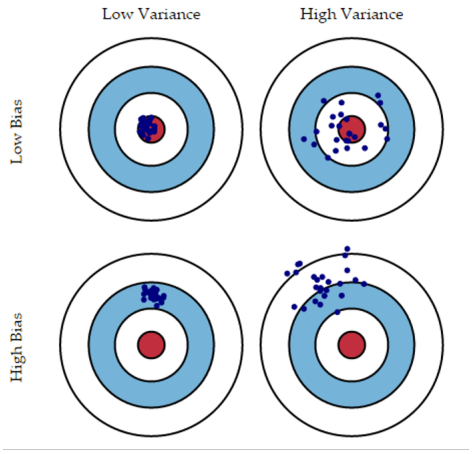
L'overfitting è un fenomeno critico in apprendimento automatico che si verifica quando un modello, addestrato eccessivamente bene sui dati di training, apprende non solo le relazioni sottostanti tra le variabili, ma anche il rumore presente nei dati. Questa eccessiva adattabilità ai dati di training impedisce al modello di generalizzare efficacemente su dati non visti, compromettendo così la sua utilità pratica. L'overfitting può essere attribuito a vari fattori, tra cui la complessità eccessiva del modello, l'eccessivo numero di parametri, o la scarsità dei dati di addestramento disponibili. In pratica, l'overfitting si manifesta con un'elevata accuratezza sui dati di training, ma con prestazioni scadenti su dati non visti o su un set di validazione.

Differenza tra bias e variance?

Il bias e la varianza sono due aspetti fondamentali dell'errore di previsione in un modello di apprendimento automatico.

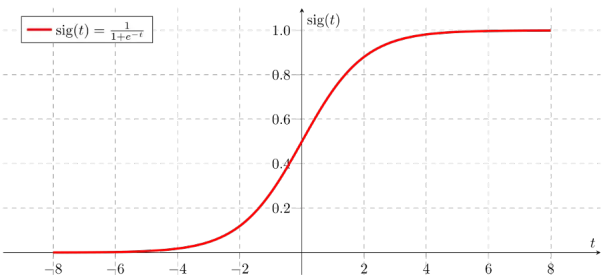
- Il bias misura quanto le previsioni del modello differiscono, in media, dal valore reale. Un alto bias generalmente indica che il modello è troppo semplice, non riuscendo a catturare la struttura sottostante dei dati, risultando in previsioni sistematicamente errate. Questo fenomeno è noto anche come underfitting.
- La varianza, invece, quantifica quanto le previsioni del modello sono sensibili a fluttuazioni nei dati di addestramento. Un modello con alta varianza è spesso troppo complesso, adattandosi eccessivamente ai dati di addestramento, inclusi il rumore e gli outlier, e risulta in un'incapacità di generalizzare bene su nuovi dati, un fenomeno noto come overfitting.

L'obiettivo nella costruzione di modelli di apprendimento automatico è trovare un equilibrio ottimale tra bias e varianza, minimizzando sia l'errore sistematico che la sensibilità alle fluttuazioni nei dati di addestramento, per costruire un modello che generalizzi efficacemente su dati non visti.



Descrivere la logistic regression

La *Regressione Logistica* è un algoritmo di apprendimento supervisionato utilizzato per problemi di classificazione binaria. Esso modella la probabilità che un'osservazione appartenga a una specifica classe utilizzando una funzione sigmoide, che mappa un input reale a un valore tra 0 e 1.



La probabilità che la variabile dipendente Y sia 1, è modellata come:

$$P(Y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_k X_k)}}$$

L'addestramento si basa sulla minimizzazione della *Log Loss*:

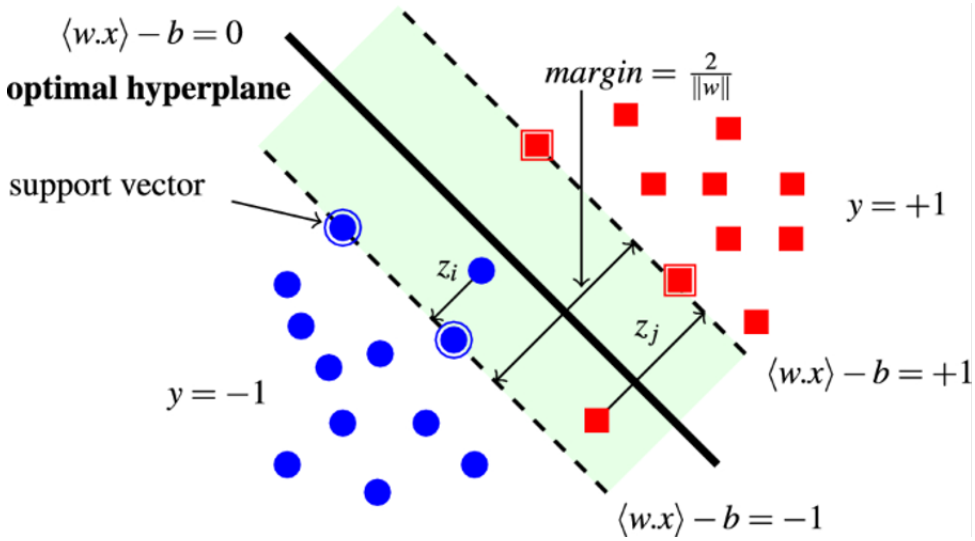
$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1-y_i)\log(1-p_i)]$$

mediante tecniche di ottimizzazione come la discesa del gradiente.

Valutazione e Interpretazione: Il modello, una volta addestrato, può essere utilizzato per classificare nuove osservazioni e i suoi coefficienti possono essere interpretati per comprendere l'importanza delle variabili indipendenti nel modello. La previsione del modello è considerata positiva quando il valore restituito dalla sigmoide è superiore a una determinata soglia (di solito 0,5), altrimenti è considerata negativa.

Limitazioni e Applicazioni: Nonostante la sua assunzione di linearità e altre limitazioni, la regressione logistica è ampiamente utilizzata in vari campi per la sua semplicità e interpretabilità.

Descrivere SVM, il suo funzionamento e le metriche



Le *Support Vector Machines* (SVM) sono uno strumento potente nel campo dell'apprendimento automatico, utilizzato principalmente per la classificazione, ma anche per la regressione. L'idea alla base di SVM è abbastanza semplice: immagina di avere dei dati che appartengono a due categorie diverse e di voler trovare la "linea" che li separa meglio.

In SVM, questa "linea" è chiamata iperpiano, e il "meglio" significa che l'iperpiano è il più lontano possibile dai punti più vicini di ogni categoria, chiamati *support vectors*. Se pensiamo ai dati come a punti in uno spazio, l'iperpiano è una sorta di "pavimento" che divide lo spazio in due parti, ognuna corrispondente a una categoria. L'iperpiano è definito come:

$$\vec{w} \cdot \vec{x} - b = 0$$

dove $\vec{w}$ è il vettore normale all'iperpiano e b è il termine di bias.

Il problema di ottimizzazione di base per un SVM è:

$$\min_{\vec{w},b} \frac{1}{2}\|\vec{w}\|^2$$

soggetto a:

$$y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1 \quad \forall i$$

dove y_i sono le etichette di classe e $\vec{x}_i$ sono i vettori di caratteristiche. Solitamente è possibile avere un margine soft o un margine hard, ovvero possiamo permetterci di fare alcuni errori di misclassificazione lasciando che magari alcune istanze oltrepassino la retta, oppure non permettere che ci siano errori nelle predizioni. Infatti, in presenza di rumore o di dati non linearmente separabili, si introduce una variabile di slack ξ_i e un parametro di regolarizzazione C per permettere alcune violazioni del margine:

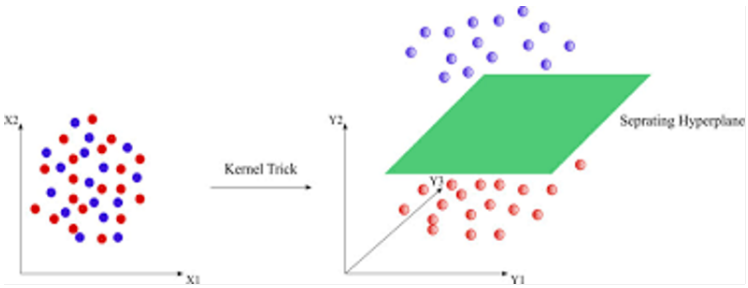
$$\min_{\vec{w},b,\xi} \frac{1}{2}\|\vec{w}\|^2 + C\sum_i \xi_i$$

soggetto a:

$$y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1 - \xi_i \quad \text{e} \quad \xi_i \geq 0 \quad \forall i$$

Kernel Trick: Per dati non linearmente separabili, SVM può essere esteso mediante l'utilizzo di funzioni kernel, che mappano implicitamente i dati in uno spazio di dimensione superiore in cui possono diventare linearmente separabili. Un kernel popolare è il kernel gaussiano (RBF):

$$K(\vec{x}, \vec{x}') = e^{-\gamma\|\vec{x}-\vec{x}'\|^2}$$

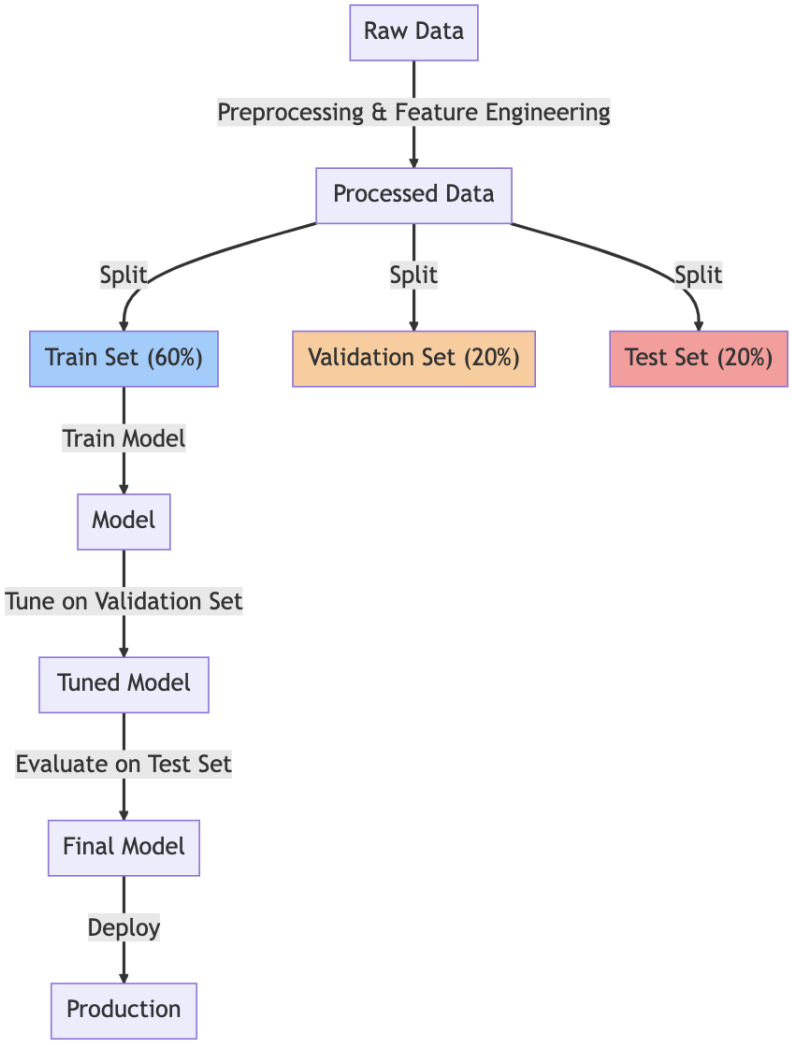


Cosa sono train validation e test

Nel campo del machine learning, il processo di sviluppo del modello richiede la divisione del dataset in tre sottoinsiemi distinti: train, validation e test. Questi sottoinsiemi hanno ruoli diversi nel processo di sviluppo e valutazione del modello.

- Train Set: Il train set è utilizzato per addestrare il modello. In questa fase, il modello apprende le relazioni e le patterns presenti nei dati, ottimizzando i suoi parametri per minimizzare l'errore nelle predizioni.
- Validation Set: Il validation set è utilizzato per valutare la performance del modello durante la fase di addestramento e per effettuare il tuning degli iperparametri. Questo set permette di identificare eventuali problemi come l'overfitting e di selezionare il modello migliore.
- Test Set: Il test set è utilizzato per valutare la performance del modello finale. Questa valutazione fornisce un'indicazione di come il modello si comporterà su dati non visti, rappresentando quindi una stima dell'errore di generalizzazione.

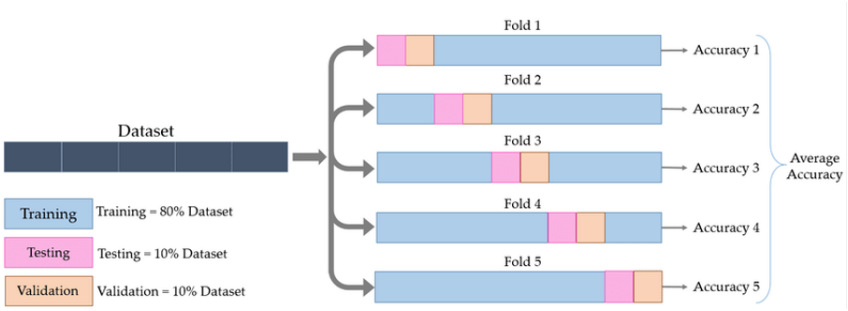
Una suddivisione comune dei dati è 60% train, 20% validation e 20% test. Tuttavia, è possibile utilizzare tecniche come la k-fold cross-validation per ottimizzare l'utilizzo dei dati disponibili, permettendo al modello di apprendere da diverse combinazioni dei dati.



Spiegare k-fold cross-validation

La k-fold cross-validation è un metodo utilizzato per valutare la performance di un modello di apprendimento automatico su un insieme di dati. Il metodo consiste nel dividere i dati in k "gruppi" di uguali dimensioni e quindi addestrare il modello su k-1 gruppi e testarlo sull'ultimo gruppo. Ciò viene ripetuto k volte, in modo che ogni gruppo venga utilizzata almeno una

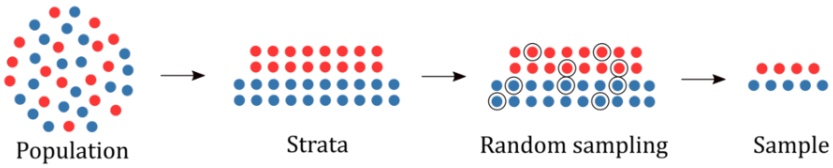
volta come set di dati di test. Alla fine, i risultati delle prove verranno combinati per ottenere una valutazione più precisa della performance del modello tramite la media.



Per esempio, se k = 5, i dati verranno divisi in 5 gruppi e il modello verrà addestrato su 4 di questi gruppi e testato sull'ultimo. Ciò verrà ripetuto 5 volte, utilizzando un gruppo diverso come set di dati di prova ogni volta. Alla fine, i risultati delle 5 prove verranno combinati per ottenere una valutazione più precisa della performance del modello tramite la media.

Il vantaggio principale della k-fold cross-validation è che utilizza tutti i dati disponibili per l'addestramento e la valutazione del modello. Ciò significa che non si perde alcun dato prezioso e si ottiene una valutazione più precisa della performance del modello rispetto ad altri metodi di validazione come la semplice divisione in training/test set. Inoltre, la k-fold cross-validation permette di valutare la robustezza del modello e di individuare eventuali problemi di overfitting o underfitting. Infatti, se un modello soffre di overfitting, i risultati sui set di dati di prova diverranno peggiori man mano che si effettuano più prove.

Cos'è lo stratified sampling e quando si utilizza



Lo stratified sampling è una tecnica di campionamento che mira a garantire che ogni sottogruppo della popolazione sia rappresentato adeguatamente nel campione finale. Questa tecnica è particolarmente utile quando la popolazione è eterogenea e presenta diverse sottopopolazioni o strati.

Nello stratified sampling, la popolazione totale viene divisa in diversi strati, o gruppi, in base a specifiche caratteristiche o attributi, come l'età, il genere, il livello di istruzione, ecc. Una volta identificati gli strati, vengono selezionati campioni casuali da ciascuno strato in modo proporzionale alla loro presenza nella popolazione totale. Questo processo garantisce che ogni strato sia rappresentato nel campione finale.

Vantaggi

- Rappresentatività: Lo stratified sampling garantisce una rappresentazione equa di tutti gli strati della popolazione, migliorando l'accuratezza e la precisione delle stime statistiche.
- Riduzione dell'Errore di Stima: La stratificazione riduce la varianza e l'errore di stima, poiché ogni strato è più omogeneo rispetto alla popolazione totale.
- Analisi Specifica degli Strati: Permette analisi più dettagliate e specifiche per ciascun strato, facilitando lo studio di sottopopolazioni specifiche.

Lo stratified sampling è particolarmente utile quando:

- dati sono eterogenei: Quando la popolazione è composta da diversi sottogruppi o strati con caratteristiche diverse.
- Presenza di sottopopolazioni di interesse: Quando è necessario analizzare specifiche sottopopolazioni o strati all'interno dei dati.
- I dati sono sbilanciati: Quando esiste uno sbilanciamento significativo tra le diverse categorie o strati nei dati, lo stratified sampling aiuta a ottenere un campione più bilanciato e rappresentativo.

Esempio: se si dispone di un dataset di pazienti e si vuole analizzare l'incidenza di una certa malattia, ma la prevalenza della malattia è diversa tra uomini e donne, si potrebbe utilizzare lo stratified sampling per garantire che il campione finale contenga un numero adeguato di uomini e donne, permettendo così analisi più accurate e affidabili.

Cosa significa fare tuning dei parametri?

Questo processo mira a trovare la configurazione ottimale degli iperparametri di un modello, ovvero quei parametri che non vengono appresi durante l'addestramento, ma che influenzano significativamente le prestazioni del modello.

L'obiettivo principale del tuning dei parametri è migliorare la capacità del modello di generalizzare su dati non visti, ottimizzando una metrica di performance specifica, come l'accuratezza, la precisione, il recall, l'F1-score, l'AUC-ROC, o la log-loss, a seconda del problema in questione.

Il tuning dei parametri si basa sull'uso di un set di validazione, su cui vengono testate diverse combinazioni di iperparametri, evitando così il rischio di overfitting sul set di test. Le principali tecniche di tuning sono:

- Grid Search: Esplora sistematicamente tutte le combinazioni possibili di valori degli iperparametri definiti a priori. È efficace ma computazionalmente costoso.
- Random Search: Esplora combinazioni casuali di valori degli iperparametri, offrendo un buon compromesso tra efficacia e costo computazionale.
- Ottimizzazione Bayesiana: Utilizza modelli probabilistici per guidare la ricerca dei valori ottimali, risultando spesso più efficiente delle tecniche precedenti.

Il tuning dei parametri è fondamentale per:

- Ottimizzare le Prestazioni: Trovare la configurazione ottimale degli iperparametri può significare la differenza tra un modello mediocre e un modello eccellente.
- Evitare Overfitting e Underfitting: Un tuning adeguato può aiutare a bilanciare la capacità del modello di apprendere dai dati e di generalizzare su nuovi dati.
- Adattare il Modello al Problema Specifico: Ogni problema ha le sue peculiarità, e il tuning permette di adattare il modello alle specifiche esigenze del problema.

Cos' è un classificatore baseline?

Un classificatore baseline, o modello baseline, è un modello semplice e fondamentale utilizzato tendenzialmente come punto di partenza nel processo di sviluppo del modello. Serve come riferimento per valutare le prestazioni di modelli più complessi e avanzati. L'obiettivo è superare le prestazioni del classificatore baseline con modelli più sofisticati.

In Task di Classificazione: In un task di classificazione, un classificatore baseline è spesso un modello che predice sempre la classe più frequente nel dataset di addestramento, indipendentemente dalle caratteristiche dell'input. Questo tipo di modello è anche conosciuto come "ZeroR" o "Majority Class Classifier".

In Task di Regressione: In un task di regressione, il classificatore baseline è generalmente un modello che predice sempre la media (o la mediana) del target nel dataset di addestramento, senza considerare il valore delle variabili indipendenti. Questo è spesso chiamato "Mean" o "Median Predictor".

Importanza del Classificatore Baseline:

- Punto di Partenza: Fornisce un punto di partenza semplice e intuitivo nel processo di modellazione.
- Benchmark: Serve come benchmark per valutare se modelli più complessi offrono miglioramenti significativi.
- Valutazione delle Prestazioni: Aiuta a stabilire un minimo livello di accettabilità per le prestazioni del modello.
- Rapidità e Semplicità: È veloce da implementare e richiede poco sforzo computazionale, permettendo una valutazione rapida del problema.

Come funziona il modello naive bayes

Si tratta di una tecnica semplice per costruire classificatori. Nonostante l'assenza di un algoritmo univoco per allenare tali classificatori, il principio comune su cui si fondano è la presupposizione che ciascuna feature sia completamente indipendente dalle altre. Questo modello prende il nome dal *Teorema di Bayes*, formalizzato come segue:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Dove:

- $P(A|B)$ è la probabilità a posteriori di A dato B.
- $P(B|A)$ è la probabilità a priori, ovvero la probabilità di osservare B dato A.

- $P(A)$ e $P(B)$ sono le probabilità marginali di A e B rispettivamente.

Nel contesto del *machine learning*, il problema si traduce nel trovare la classe C_k che massimizza la probabilità a posteriori, ovvero:

$$C_k = \arg\max_k P(C_k|\text{feature}_1, \text{feature}_2, \ldots, \text{feature}_n)$$

Sotto l'assunzione di indipendenza condizionale tra le features, si ottiene:

$$P(\text{feature}_1, \text{feature}_2, \ldots, \text{feature}_n|C_k) = \prod_{i=1}^{n} P(\text{feature}_i|C_k)$$

Pertanto, la predizione del modello sarà la classe che massimizza la probabilità a posteriori. Questo è fondamentalmente l'approccio adottato dai classificatori Naive Bayes nel *machine learning*.

Spiegare il criterio di splitting basato su Information Gain Index

DA FARE

Spiegare i metodi principali per processare testo

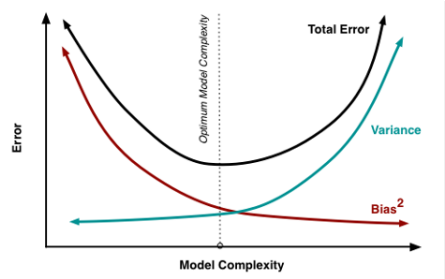
Lo scopo del text-processing nasce dalla necessità di poter salvare e computare input testuali con il minor numero di risorse possibili, mantenendo le stesse informazioni.

Alcuni metodi per fare ciò sono:

- Tokenization: si tratta il testo come una serie di token, che possono essere parole frasi o paragrafi;
- Lemming: si rimuovono i suffissi
- STemming: si mantiene la forma base delle parole, previa analisi morfologica;
- Rimozione delle "stopword": si rimuovono tutti i termini che rappresentano punteggiature, congiunzioni, articoli e preposizioni che non hanno un contributo significativo al valore del testo;
- Normalization: elimina sistematicamente lettere maiuscole e/o segni di punteggiatura dal testo, per ridurre il numero di caratteri che si possono incontrare;
- Aggiunta di informazioni semantiche: in pratica come l'identificazione di entità, relazioni o azioni presenti nel testo

Un altra tecnica è l'impiego di shingling, min-hashing o locally sensitive hashing (LSH) spiegato nelle domande

Spiegare bias-variance decomposition



La *decomposizione di bias-variance* è fondamentale in apprendimento automatico, permettendo di analizzare l'errore di previsione di un modello. L'errore totale può essere scomposto in *bias*, *varianza*, ed *errore irriducibile*.

Componenti dell'Errore

- **Bias:** Misura la differenza tra le previsioni del modello e i valori reali. Alto bias causa *underfitting*.
- **Varianza:** Misura le variazioni delle previsioni per diversi insiemi di dati di addestramento. Alta varianza causa *overfitting*.
- **Errore Irriducibile:** Errore intrinseco, dovuto al rumore nei dati, che non può essere ridotto.

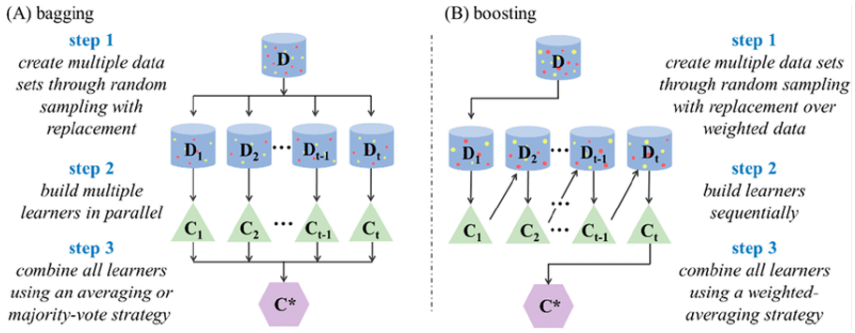
Formula dell'Errore Totale:
L'errore totale E può essere rappresentato come:

$$E = \text{Bias}^2 + \text{Varianza} + \text{Errore Irriducibile}$$

Trade-off Bias-Varianza: Esiste un trade-off tra bias e varianza. L'obiettivo è minimizzare l'errore totale attraverso tecniche come la regolarizzazione e l'ottimizzazione degli iperparametri, bilanciando bias e varianza.

Soluzioni:
Le soluzioni includono la selezione delle caratteristiche, la regolarizzazione, e l'ottimizzazione degli iperparametri per trovare un equilibrio ottimale e ridurre l'errore complessivo.

Spiegare quando è utile usare bagging(descriverlo) e quando invece è utile usare boosting(descriverlo)



Bagging e Boosting sono due tecniche di ensemble utilizzate per migliorare la performance di un modello di apprendimento automatico.

- Bagging: Il Bagging è una tecnica che consiste nel creare più copie del modello originale, ognuna addestrata su un sottoinsieme casuale dei dati di addestramento. Il risultato finale è un insieme di modelli che possono essere utilizzati insieme per fare previsioni, come la media o la moda delle previsioni dei singoli modelli in base al task richiesto. È particolarmente utile per modelli ad alto rischio di overfitting, come gli alberi di decisione profondi.
- Boosting: Il Boosting è una tecnica che consiste nell'addestrare una serie di modelli in sequenza, dove ogni modello cerca di correggere gli errori del modello precedente. Il risultato finale è un insieme di modelli che lavorano insieme per fare previsioni, come la somma ponderata delle previsioni dei singoli modelli. È particolarmente utile per modelli a basso rischio di overfitting, come i modelli lineari.

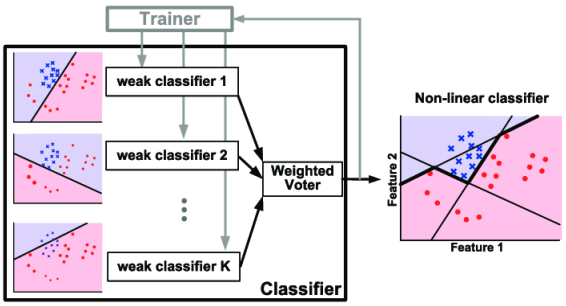
Scelta della Tecnica: La scelta tra bagging e boosting dipende dalla natura del problema e dalla tipologia del modello. Se il modello soffre di alta varianza, il bagging potrebbe essere la scelta migliore. Se il modello ha un alto bias, il boosting potrebbe essere più appropriato.

Compromesso Bias-Varianza: Entrambe le tecniche cercano di ottimizzare il compromesso bias-varianza, ma lo fanno in modi diversi. Il bagging mira a ridurre la varianza senza aumentare il bias, mentre il boosting mira a ridurre il bias senza aumentare eccessivamente la varianza.

Spiegare adaboost

AdaBoost, o Adaptive Boosting, è un algoritmo di boosting particolarmente efficace e popolare. L'obiettivo di AdaBoost è di combinare i punti di forza di molti modelli deboli per creare un modello forte ed accurato. Qui di seguito è spiegato più dettagliatamente il funzionamento di AdaBoost:

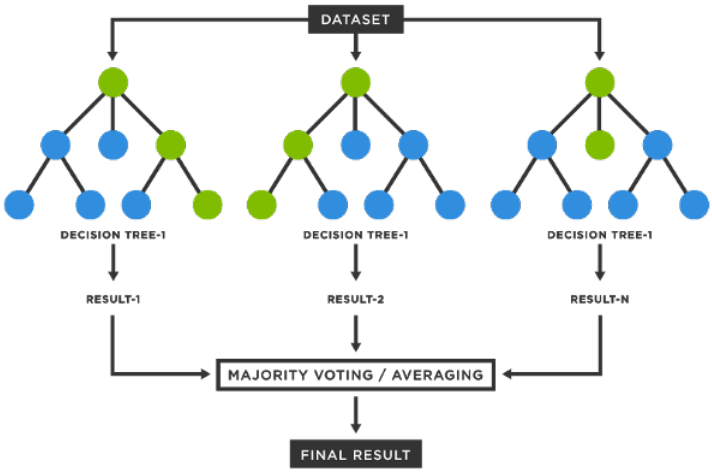
1. Inizializzazione dei Pesi: Ogni osservazione nel dataset inizialmente ha lo stesso peso, $1/n$, dove n è il numero totale di osservazioni.
2. Creazione del Modello Debole: AdaBoost crea un modello debole, solitamente un albero di decisione con un solo livello (stump).
3. Calcolo dell'Errore: L'errore del modello è calcolato come la somma dei pesi associati alle osservazioni classificate in modo errato.
4. Calcolo dell'Importanza del Modello: AdaBoost calcola l'importanza del modello debole in base al suo errore.
5. Aggiornamento dei Pesi: I pesi delle osservazioni sono aggiornati in modo che le osservazioni classificate in modo errato ricevano più peso, mentre quelle classificate correttamente ricevono meno peso.
6. Iterazione: Il processo è ripetuto, creando e aggiungendo modelli deboli al modello complessivo finché non si raggiunge un numero predeterminato di modelli o finché il modello complessivo non classifica perfettamente il training set.
7. Creazione del Modello Finale: Il modello finale è una combinazione ponderata dei modelli deboli, in cui il peso di ogni modello è determinato dalla sua accuratezza.



Vantaggi di AdaBoost:

- È in grado di ridurre il bias e la varianza.
- È efficace con dataset sbilanciati.
- Può essere utilizzato con vari tipi di modelli di apprendimento.

Spiegare random forest

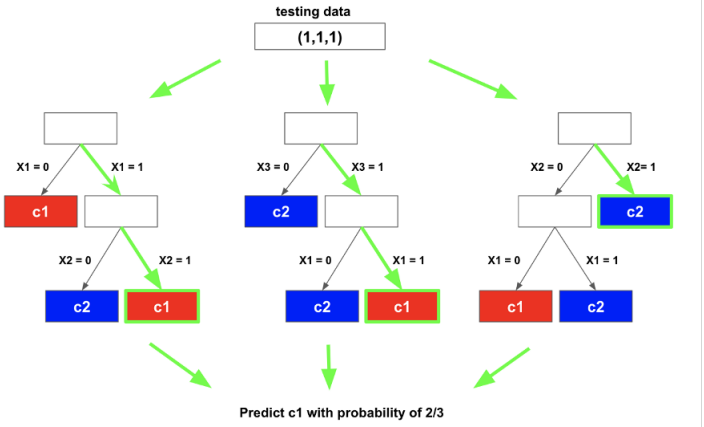


Random Forest è un algoritmo molto utilizzato di bagging, ovvero si utilizza per cercare di abbassare la varianza.
L'idea generale di un algoritmo di boosting solitamente è:

1. si prende il dataset originale e da questo si estrae tramite la tecnica di bootstrap un campione (campionamento con rimpiazzamento) e la dimensione del campione pescato è uguale alla dimensione del dataset iniziale;
2. Dopo si allenano i diversi modelli in parallelo sui vari campioni pescati e la predizione sarà in base al task che dobbiamo risolvere: una media di tutte le previsioni dei vari modelli nel caso di regressione e la moda nel caso invece di classificazione.

Ciò per cui differisce la random forest è che è importante che i modelli siano il più possibile indipendenti, per questo motivo random forest oltre a fare il procedimento descritto, durante la creazione di questi alberi che devono essere fully grown (con basso bias e elevata varianza) si cerca di renderli il più diversi possibili utilizzando la random input selection ovvero si utilizza un sottoinsieme di features diverse su ogni modello che si allena in modo tale che ogni albero sia il più possibile diverso dagli altri

Come è possibile usare la random forest per stimare la similarità



Per stimare la similarità tra due istanze con una Random Forest, si osserva in quali foglie delle diverse alberi le due istanze finiscono. Se due istanze finiscono frequentemente nelle stesse foglie attraverso i diversi alberi della foresta, si può inferire che sono simili tra loro. Questo perché le istanze che percorrono gli stessi cammini e finiscono nelle stesse foglie hanno caratteristiche simili, secondo i criteri di divisione degli alberi.
Procedura:

- Addestramento della Foresta: Addestra una Random Forest sul tuo set di dati.
- Percorso delle Istanze: Per ogni istanza, registra le foglie in cui cade in ogni albero della foresta.
- Calcolo della Similarità: Per ogni coppia di istanze, calcola la percentuale di alberi in cui cadono nella stessa foglia. Una percentuale più alta indica una maggiore similarità.

Questa misura di similarità può essere utilizzata in vari contesti, come il clustering, la riduzione della dimensionalità, o come feature in altri modelli di machine learning.
Vantaggi

- Robustezza: La Random Forest è resistente agli outlier e può gestire bene le variabili categoriche e continue.
- Interpretabilità: La similarità basata sulla Random Forest può fornire intuizioni intuitive, poiché si basa sulla struttura degli alberi di decisione.

Esempio: considera due istanze A e B. Se, in una Random Forest di 100 alberi, A e B cadono nella stessa foglia in 85 alberi, lo score di similarità sarà 85%.

Come è possibile usare la random forest per identificare gli outliers?

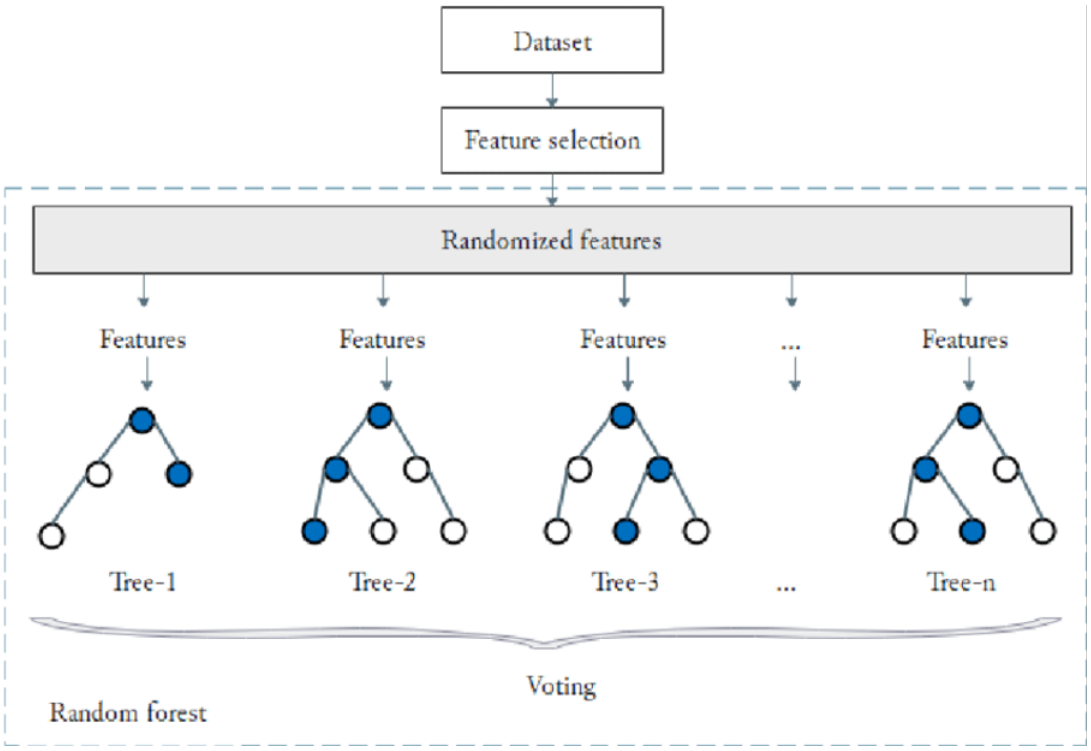
Per identificare gli outliers si può utilizzare un procedimento simile a quelli visto sopra ovvero dato un punto per capire se è un outlier si può misurare il suo score come outlier misurando la sua dissimilarità dal resto dei punti come 1 / la similarità calcolata dalla random forest, ovviamente più sarà alto lo score più sarà probabile che sia un outlier.

Come è possibile usare la random forest per rimpiazzare i valori mancanti?

È possibile usare una random forest per rimpiazzare i valori mancanti infatti grazie ad essa possiamo per un valore mancante di una determinata feature trovarlo.
Il procedimento descritto brevemente si basa su calcolare la similarità basata sulla random forest dell'istanza dove manca il valore di una determinata feature con tutto il resto delle istanze e per ognuna di queste si prende dove è presente il valore della future che vogliamo sistemare e si fa una media pesata sulla base della similarità calcolata dalla random forest.

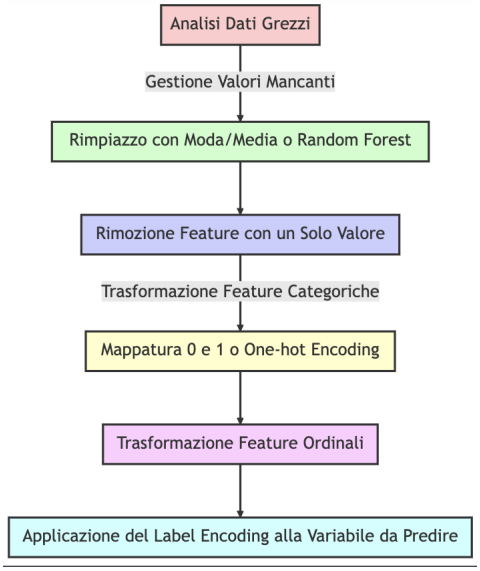
Questo procedimento può essere iterato più volte sui vari valori mancanti del dataset. Solitamente un altro procedimento utilizzato è quello di rimpiazzare un valore numerico con la media dei valori che assume quella feature oppure se categorico la moda.

Come è possibile fare feature selection con la random forest?

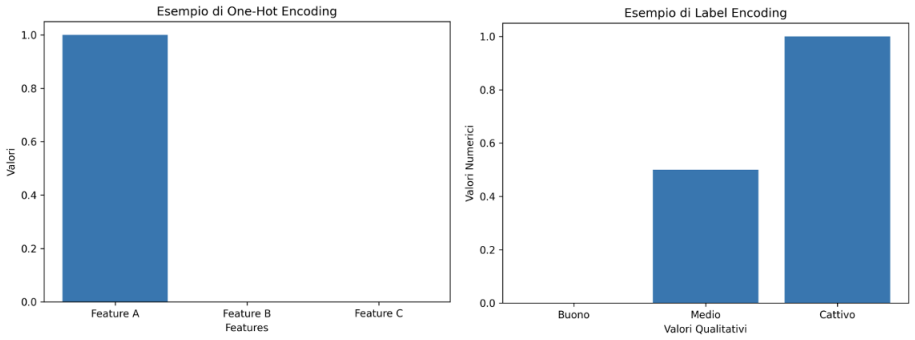


È possibile inoltre utilizzare la random forest per fare feature selection e quindi evitare il problema della dimensionalità. Infatti grazie alla random forest, è possibile salvare durante la fase di creazione dei diversi alberi quali sono stati gli split (feature e treshold) che hanno portato a un maggior guadagno, in tal modo poi è possibile ordinare le varie feature in base proprio all'importanza determinata dal gain a splittare, e sulla base di questo le feature meno importanti si può decidere di scartare le feature meno importanti e di riapplicare il procedimento più volte fino a che non abbiamo ottenuto un numero di feature ridotto.

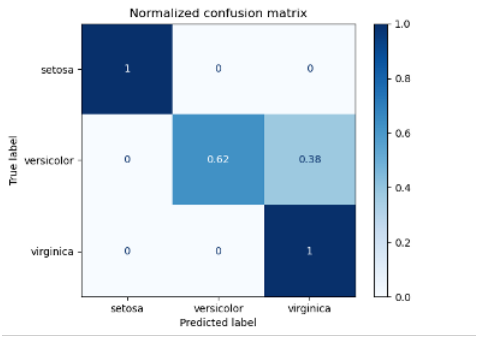
Come si esegue il feature engineering



Il feature engineering è una delle parti più importanti durante la creazione di un modello predittivo. È la fase in cui dai dati raccolti (grezzi) si analizzano e si trasformano cercando di tenere solamente quelli che si ritengono importanti e rimuovendo gli outliers. Innanzitutto solitamente si osserva se sono presenti valori mancanti nel dataset e solitamente vengono rimpiazzati con la moda nel caso sia una feature categorica o con la media se è una feature numerica (se si vuole è anche possibile utilizzare una random forest per stimare i valori mancanti).
Dopo aver sistemato i valori mancanti si osserva se ci sono feature contenenti solo un valore: in questo caso si possono scartare perché non forniscono informazioni interessanti, e poi si passa alla trasformazione delle feature categoriche dato che la maggior parte degli algoritmi di machine learning funzionano bene con valori numerici.
Se la feature è categorica e ha solamente due valori si può rimappare tra 0 e 1, altrimenti se ha k valori si può utilizzare la tecnica del one-hot encoding ovvero per ogni valore che assume la feature categorica che vogliamo trasformare viene aggiunta una colonna binaria dove con 1 indichiamo la presenza di quel valore e con 0 l'assenza. Inoltre sono presenti anche le feature ordinali che assumono ad esempio valori come (buono, medio, cattivo) solitamente queste vengono rimappate numericamente mantenendo l'ordine qualitativo ad esempio (0, 0.5, 1).
Inoltre per la variabile da predire se è categorica si può applicare il label encoding, ovvero si rimappa in id numerici.



Cos'è la confusion matrix



La Confusion Matrix è un elemento chiave nella valutazione dei modelli di classificazione nell'apprendimento supervisionato. Essa fornisce una rappresentazione tabulare delle performance del modello, confrontando le classi reali con quelle predette. In una matrice binaria, si distinguono True Positive (TP), True Negative (TN), False Positive (FP) e False Negative (FN), che rappresentano rispettivamente le classificazioni corrette e quelle errate di ciascuna classe.

In una classificazione multiclasse, la matrice si espande per includere queste categorie per ogni classe. La diagonale principale rappresenta le classificazioni corrette, mentre gli elementi fuori dalla diagonale indicano errori. Un modello ideale avrebbe solo valori non nulli sulla diagonale.

Dalla Confusion Matrix si derivano metriche cruciali come Accuratezza, Precisione, Recall e F1 Score, che permettono di valutare l'efficacia del modello. Questa matrice è particolarmente utile per identificare le aree di debolezza del modello e ottimizzare le sue performance, essendo rappresentabile anche visivamente attraverso un heatmap, facilitando così l'interpretazione dei risultati.

Quali sono le principali metriche di prestazione di un modello

Le metriche per stabilire le prestazioni di un modello dipendono fortemente dal tipo di task che tale modello deve portare a termine.

- Classificazione:
  - **Accuracy**: La percentuale di predizioni corrette prodotte dal modello.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

    Esempio: Se il modello ha correttamente predetto 80 su 100 campioni, l'accuracy è del 80%.
  - **Precision**: Indica la proporzione di identificazioni positive che sono effettivamente corrette.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

    Esempio: Se il modello ha identificato 50 positivi, ma solo 40 sono veri positivi, la precision è 0.8.
  - **Recall**: Mostra la proporzione di effettivi positivi che sono stati identificati correttamente.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

    Esempio: Se ci sono 50 positivi reali e il modello ne ha identificati 40, il recall è 0.8.
  - **F1-Score**: Media armonica tra Precision e Recall.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

    Esempio: Con Precision=0.8 e Recall=0.8, l'F1-Score è 0.8.
  - **AUC-ROC**: Area sotto la curva ROC. Un valore di 1.0 indica una classificazione perfetta.(vedi domanda successiva)
  - **Log-Los**: Misura la performance di un modello di classificazione.

$$\text{Log-Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))]$$

    Esempio: Un modello con log-loss inferiore ha migliori performance.
- Regressione:

– **MSE**: Calcola la media dei quadrati degli errori.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Esempio: Differenza tra i valori osservati e quelli predetti.

– **R-Squared**: Indica la percentuale di varianza della variabile dipendente spiegata dal modello.

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}}$$

Esempio: Un R^2 di 0.8 indica che l'80% della varianza è spiegata dal modello.

Spiegare cos'è la ROC curve (e AUC) / Come si valuta un classificatore binario

La curva ROC (Receiver Operating Characteristic) è uno strumento grafico fondamentale per valutare la capacità di un modello di classificazione binaria di distinguere tra le classi positive e negative. Essa traccia il Tasso di Veri Positivi (True Positive Rate, TPR) contro il Tasso di Falsi Positivi (False Positive Rate, FPR) a vari livelli di soglia di classificazione.
TPR (Sensibilità o Recall): È la proporzione di osservazioni positive reali che sono correttamente identificate dal modello.

True Positive Rate (TPR) o Sensibilità:

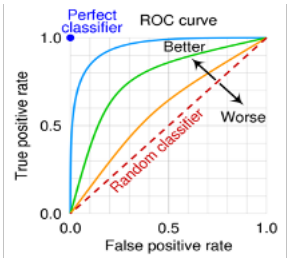
$$TPR = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Dove TP sono i Veri Positivi e FN sono i Falsi Negativi. Rappresenta la proporzione di osservazioni positive reali che sono correttamente identificate dal modello. FPR (1 - Specificità): È la proporzione di osservazioni negative reali che sono erroneamente identificate come positive.

False Positive Rate (FPR) o $1 -$ Specificità:

$$FPR = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

Dove FP sono i Falsi Positivi e TN sono i Veri Negativi. Rappresenta la proporzione di osservazioni negative reali che sono erroneamente identificate come positive.



La linea diagonale rappresenta la performance di un classificatore casuale; un modello utile si trova sopra di essa.
Un punto in alto a sinistra del grafico indica un basso FPR e un alto TPR, il che è ideale.
L'area sotto la curva ROC, o AUC, è un indicatore della qualità del modello. Un AUC di 1.0 indica un modello perfetto, mentre un AUC di 0.5 indica un modello non informativo, equivalente a un lancio di moneta.

L'AUC è un indicatore numerico della capacità del modello di distinguere tra classi positive e negative. Valori di AUC più vicini a 1 indicano un migliore equilibrio tra Sensibilità e Specificità, mentre un AUC di 0.5 suggerisce che il modello non ha capacità discriminante.

Valutazione del Classificatore:
AUC elevata: Indica che il modello ha una buona capacità di distinguere tra le classi. AUC bassa: Suggerisce che il modello ha difficoltà a distinguere tra le classi. AUC = 0.5: Il modello non è in grado di distinguere tra le classi, equivalente a un classificatore casuale. AUC = 1.0: Il modello è in grado di classificare perfettamente tutte le osservazioni. Esempio: Un modello con un AUC di 0.8 è generalmente considerato buono, ma potrebbe comunque beneficiare di ottimizzazioni per ridurre ulteriormente gli errori di classificazione. Un modello con un AUC di 0.9 o superiore è considerato eccellente.

Overfitting vs underfitting

Overfitting si verifica quando un modello di apprendimento automatico è troppo complesso rispetto ai dati di addestramento e inizia a memorizzare i dettagli delle singole osservazioni, invece di generalizzare le tendenze generali. Ciò può causare un alto rendimento sui dati di addestramento, ma un rendimento scarso sui dati di test o di validazione.
Underfitting si verifica quando un modello è troppo semplice rispetto ai dati di addestramento e non è in grado di catturare le tendenze nei dati. Ciò può causare un basso rendimento sia sui dati di addestramento che sui dati di test o di validazione.
In generale, il modello ideale dovrebbe essere complesso abbastanza da catturare le tendenze nei dati, ma non così complesso da memorizzare i dettagli delle singole osservazioni.

A cosa servono le association rules

Lorem ipsum

Spiegare apriori

Lorem ipsum

Spiegare fp growth

Lorem ipsum

Spiegare cos'è k-shingles

K-Shingles è una tecnica che permette di trovare documenti simili a una query di documenti in modo efficiente sia in termini di tempo che di spazio, superando la necessità di una corrispondenza esatta delle stringhe. Questo metodo è particolarmente utile per confrontare la similarità tra documenti basandosi sulla sintassi.
Un documento è considerato come una stringa di caratteri, e i suoi k-shingles sono definiti come l'insieme di tutte le possibili sottostringhe di lunghezza k presenti nel documento. La similarità tra due documenti può quindi essere calcolata utilizzando la similarità di Jaccard tra i loro shingles.

Calcolo della Similarità: La similarità di Jaccard tra due insiemi è definita come:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

dove A e B sono gli insiemi di shingles dei due documenti.
Scelta del Valore di K: La scelta del valore di k è cruciale:

- Valori bassi di k aumentano la probabilità di trovare shingles comuni, aumentando la similarità.
- Valori alti di k aumentano il potere discriminante dei shingles.
- Un valore di k compreso tra 5 e 9 è spesso considerato ottimale, a seconda della lunghezza dei documenti.

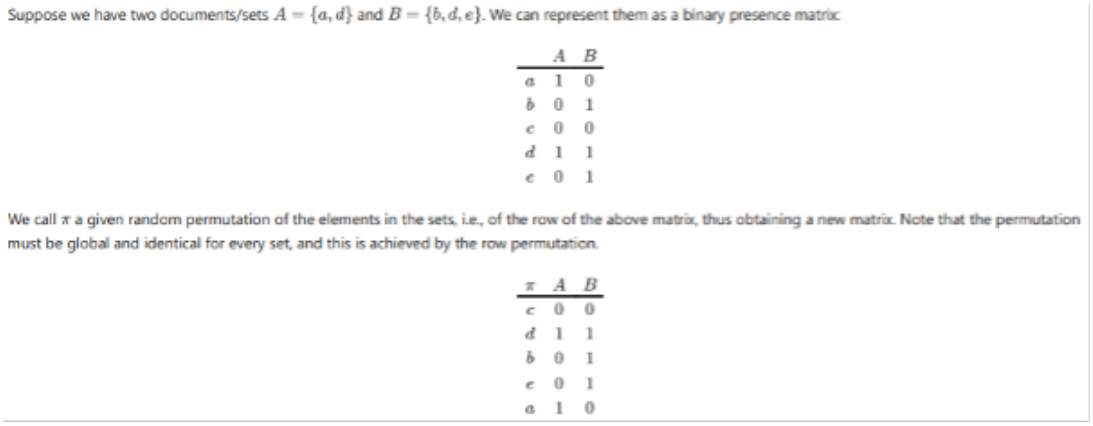
Numero di Shingles: Il numero di shingles per un documento d è calcolato come:

$$\#\text{shingles}(d) = |d| - k + 1$$

dove $|d|$ è la lunghezza del documento in caratteri.
Ottimizzazione: Data la potenziale onerosità computazionale della strategia di k-shingles, sono state sviluppate tecniche di ottimizzazione come il Min-Hashing e il Locality-Sensitive Hashing (LSH) per ridurre ulteriormente il costo computazionale nella comparazione di documenti.

Spiegare il Min-Hashing

La *Min-Hashing* è una tecnica avanzata utilizzata per rappresentare in modo efficiente e compatto i documenti, mantenendo la capacità di calcolare accuratamente la similarità tra di essi. Questa tecnica è particolarmente utile quando si desidera trovare documenti simili a un documento di query in modo efficiente sia in termini di tempo che di spazio, superando i limiti della corrispondenza esatta delle stringhe.
Dati due documenti, A e B, il min-hash di A è definito come il più piccolo elemento dell'insieme A dopo l'applicazione di una permutazione casuale π. Importante notare che questa permutazione è globale e applicata in modo uniforme a tutti i documenti nell'insieme di dati.



Suppose we have two documents/sets $A = \{a, d\}$ and $B = \{b, d, e\}$. We can represent them as a binary presence matrix:

	A	B
a	1	0
b	0	1
c	0	0
d	1	1
e	0	1

We call π a given random permutation of the elements in the sets, i.e., of the row of the above matrix, thus obtaining a new matrix. Note that the permutation must be global and identical for every set, and this is achieved by the row permutation.

π	A	B
c	0	0
d	1	1
b	0	1
e	0	1
a	1	0

Matematicamente, se definiamo $\min(\pi A) = d$ e $\min(\pi B) = d$, allora il min-hash fornisce un mezzo per calcolare la similarità in quanto la probabilità che i due min-hashes siano uguali è equivalente alla similarità di Jaccard dei due documenti, ovvero:

$$P(\min(\pi A) = \min(\pi B)) = J(A, B)$$

Questa tecnica offre vantaggi significativi rispetto al metodo k-shingles. Mentre salvare un singolo shingle è certamente più efficiente dal punto di vista della memoria rispetto a salvare l'intero documento, il metodo k-shingles non fornisce informazioni sulla misura in cui due documenti non simili differiscono. Questa lacuna informativa può essere colmata utilizzando la tecnica di min-hash in combinazione con la *Local Sensitivity Hashing (LSH).*

Spiegare Min-Hash Signatures e LSH

LOOK HERE
Min-Hash Signatures e *LSH* (Locality Sensitive Hashing) sono tecniche avanzate utilizzate per superare le limitazioni della tecnica min-hashing, permettendo di calcolare efficientemente la similarità tra documenti. Quando due documenti non sono simili, la tecnica min-hashing non fornisce informazioni sulla misura in cui differiscono. Per superare questa limitazione, si possono utilizzare più min-hash.
Dando m permutazioni, $\pi_1, \pi_2, \ldots, \pi_m$, per ogni documento A si possono calcolare gli m min-hashes di tali permutazioni ottenendo il vettore

$$\min(\pi A_1), \min(\pi A_2), \ldots, \min(\pi A_m)$$

, cioè la min-hash signature. Questa signature può essere usata per stimare la similarità tra due documenti A, B con la formula:

$$J(A, B) \approx \frac{k}{m}$$

dove k è il numero di min-hashes corrispondenti ottenuti da m permutazioni, e $J(A, B)$ è la distanza di Jaccard tra i documenti A, B.
LSH interviene per risolvere il problema della ricerca e selezione veloce, dividendo gli m min-hashes in b gruppi con $r = \frac{m}{b}$ elementi ognuno. Ogni sottosequenza di r min-hashes viene concatenata e sottoposta nuovamente a hash per ottenere un nuovo hash noto come super-signature (o super-shingle). Queste super-signatures sono usate come chiavi per accedere alla tabella hash.

Spiegare sim-hashing

Sim-Hashing è una tecnica avanzata che permette di rappresentare documenti in uno spazio multidimensionale e di calcolare la similarità del coseno tra di loro. Dati due documenti, rappresentati in uno spazio bidimensionale, essi sono separati da un angolo θ. Utilizzando un vettore random r (o un iperpiano r in spazi di dimensione maggiore), si possono calcolare le probabilità che i due documenti cadano dalla stessa parte di r e che r cada tra i due documenti, permettendo così di stimare la cosine-similarity tra i documenti.
L'algoritmo di Sim-Hashing funziona come segue:

1. Scegliere randomicamente r come un vettore o iperpiano disegnato uniformemente.
2. Dato un documento A, calcolare il prodotto scalare $r \cdot A$ per determinare su quale lato dell'iperpiano A si trova, assegnando 1 se positivo, altrimenti 0.
3. Ripetere i passi da 1 a 2 per m volte per calcolare una firma di m bit.
4. Dati due documenti, calcolare la distanza di Hamming tra le loro signatures per stimare il loro angolo e quindi la cosine similarity.

Spiegare le misure di qualità dei recommender systems

I *Recommender Systems* sono sistemi di filtraggio delle informazioni che mirano a prevedere la preferenza dell'utente e suggerire prodotti, servizi o informazioni pertinenti. La loro qualità è valutata secondo diverse misure:

1. **Efficienza nella costruzione del modello:** Valuta il costo computazionale associato al processamento dei dati e alla costruzione del Recommender System, includendo l'analisi necessaria per generare i dati utilizzati.
2. **Efficienza nella generazione dei suggerimenti:** Misura il costo computazionale delle raccomandazioni a run-time, ovvero il tempo e le risorse necessarie per generare suggerimenti in tempo reale.
3. **Serendipità delle raccomandazioni:** Le raccomandazioni devono essere nuove, non banali, e inaspettate, aiutando gli utenti a scoprire nuovi oggetti e ad esplorare nuovi interessi.
4. **Adattamento al problema della "partenza a freddo":** Valuta come il modello si comporta quando incontra nuovi utenti o nuovi items, per i quali ha poche o nessuna informazione precedente.

Queste misure permettono di valutare l'efficacia e l'efficienza dei Recommender Systems, garantendo che siano in grado di fornire suggerimenti pertinenti e utili agli utenti, anche in presenza di informazioni limitate.

Spiegare le varie tecniche usate in recommender system

Lorem ipsum

Quali sono le principali misure di similarità?

Lorem ipsum

Spiegare il problema della dimensionalità

Lorem ipsum

Come è possibile risolvere il problema della dimensionalità tecniche

Lorem ipsum

Discutere come stimarela distanzadi Jaccard tradue insieme tramite MinHashing

Lorem ipsum

Discuterebrevementecosasiintendeper dimensionality curse

Lorem ipsum

Discuterela rappresentazionevettorialedeidocumentibasatato tfidf

Cosa significa fare clustering

Lorem ipsum

Descrivere i vari tipi di clustering

Lorem ipsum

Approcci al clustering

Lorem ipsum

Come funziona k-means

Lorem ipsum

Come funziona k-means++

Lorem ipsum

Come funziona k-medoid (PAM)

Lorem ipsum

Spiegare come funziona HAC e le varie misure (vedi anche complessità)

Lorem ipsum

A cosa serve il dendrogramma

Lorem ipsum

Spiegare clustering divisivo

Lorem ipsum

Spiegare DBScan

Lorem ipsum

Spiegare la valutazione del clustering intrinseca ed estrinseca / Cos'è il silhouette coefficient

Lorem ipsum

Cos'è l'hard clustering e il soft clustering?

Lorem ipsum

Spiegare fuzzy C-means.

Lorem ipsum

Spiegare SOM(self organizing map)

Lorem ipsum

Spiegare cos'è un ANN e come funziona

Lorem ipsum

Spiegare le varie funzioni di attivazione

Lorem ipsum

Cos'è la loss function per un ANN

Lorem ipsum

Cosa apprende un ANN

Lorem ipsum

Cos'è una rete convoluzionale e come funziona

Lorem ipsum

A cosa servono i filtri

Lorem ipsum

Per quali task può essere usata un ANN e invece una convolutional network?

Lorem ipsum

Come risolvere l'overfitting in un ANN

Lorem ipsum

Vedi web search and ranking

Lorem ipsum

Raccolta esercizi pratici

Dato un dataset, trovare la radice del DT usando GINI Index

Lorem ipsum

Dato un dataset, trovare la radice del DT usando Information Gain

Lorem ipsum

Dato il seguente training set, usare un classificatore Bayesiano per predire la classe"PlayTennis" nel test set

Lorem ipsum

Dato un dataset, predire la classe di decisione per la nuova istanza specificata

Lorem ipsum

Data una matrice Transaction ID - Items, trovare l'item set di candidati X con A-Priori Algorithm, considerando 2 come supporto minimo. Calcolare anche la confidenza degli elementi dei candidati risultanti, con confidenza minima pari al 60%

Lorem ipsum

Data una matrice Transaction ID - Items, trovare il Frequent Pattern con l'algoritmo FP_Growth

Lorem ipsum

Filtri convoluzionali

Lorem ipsum