Data and Web Mining

Raccolta domande teoriche

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

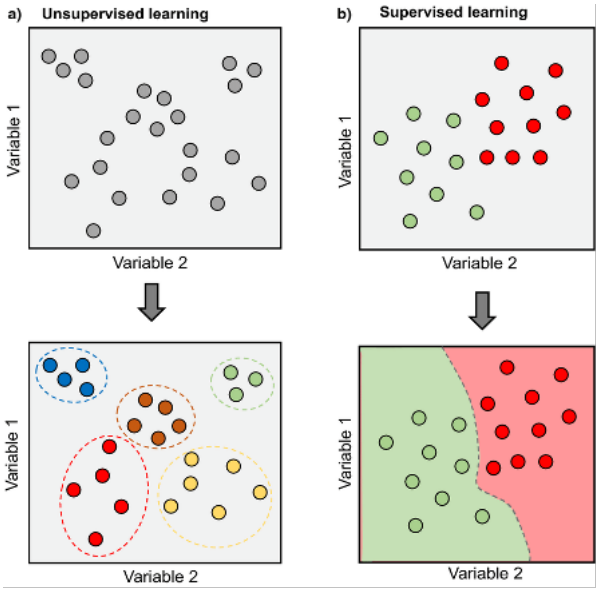
Supervised Learning:

- 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.

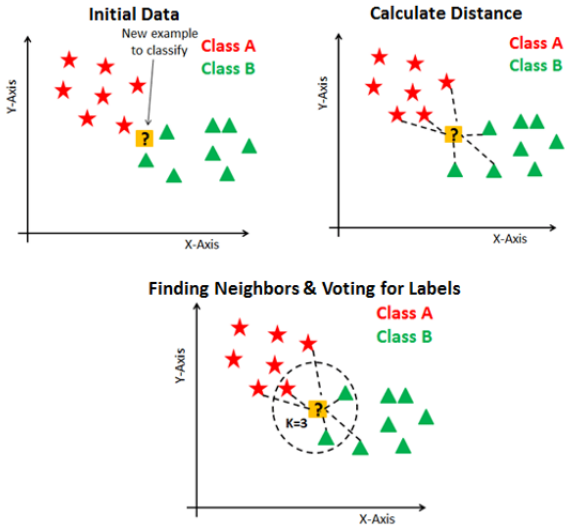
In generale, gli algoritmi di apprendimento supervisionato sono utilizzati per problemi di classificazione e regressione, mentre gli algoritmi di apprendimento non supervisionato sono utilizzati per problemi di clustering e riduzione della dimensionalità.



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:

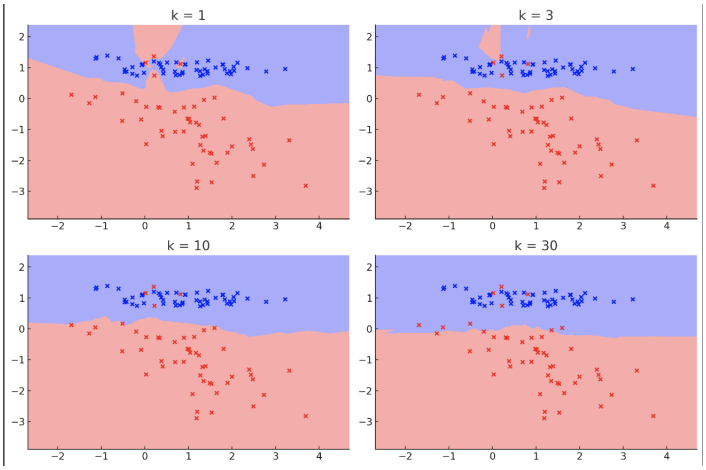
- Initial Data: Si parte con un dataset di addestramento dove ogni punto è etichettato con una classe o un valore.
- Calculate Distance: Per ogni nuovo punto, si calcola la distanza da tutti i punti nel dataset di addestramento. La distanza euclidea è comunemente utilizzata, ma altre metriche di distanza 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.



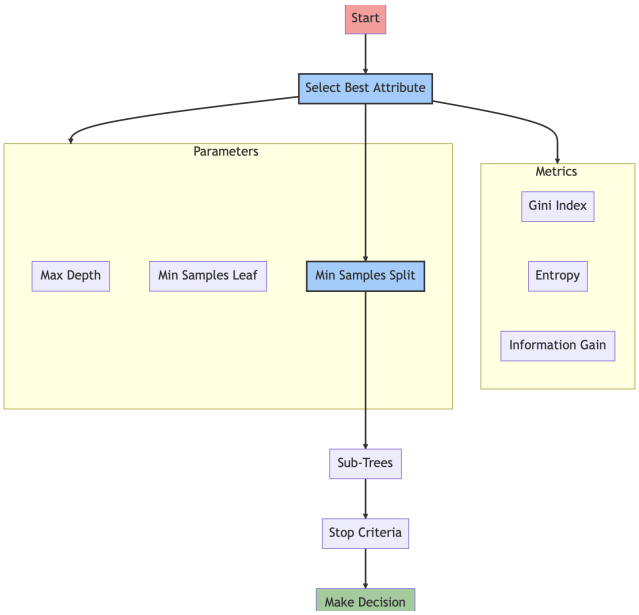
L'immagine mostra come l'algoritmo K-nn varia con diversi valori di k, illustrando l'effetto di overfitting con k piccoli e di underfitting con k grandi. Nello specifico abbiamo ogni grafico che rappresenta un diverso valore di k e mostra come l'algoritmo classifica i punti in due classi diverse (colori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

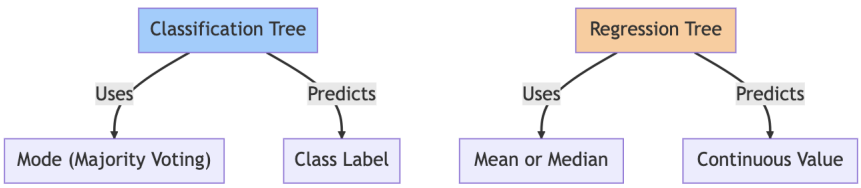
- **Classification Error:**
$$\text{Error} = 1 - \max(p_1, p_2, \ldots, p_m)$$
dove p_i rappresenta la proporzione di campioni appartenenti alla classe i in un nodo.
- **Information Gain:**
$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Val}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$
dove 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.



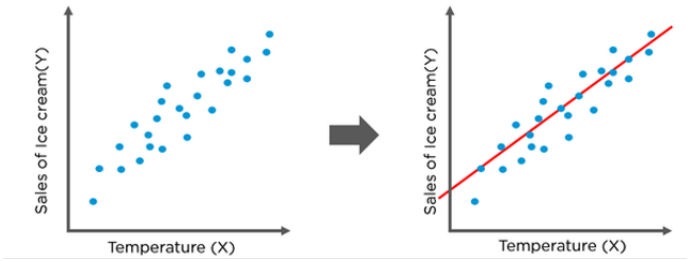
Descrivere un algoritmo efficiente per costruire un decision Tree

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:

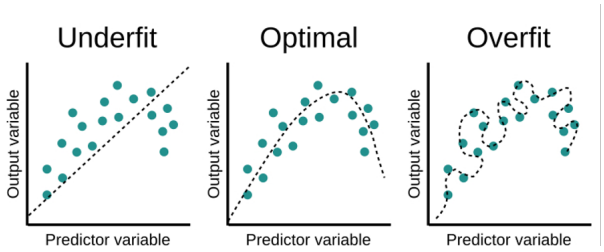
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

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?



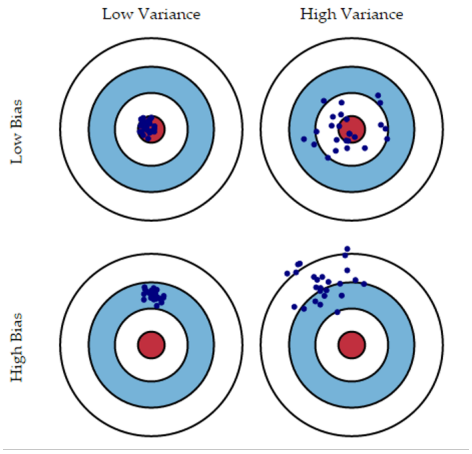
L'overfitting è un fenomeno critico in apprendimento automatico che si verifica quando un modello, addestrato eccessivamente bene sui dati di training, apprende non solo le relazioni sottostanti tra le variabili, ma anche il rumore presente nei dati. Questa eccessiva adattabilità ai dati di training 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.

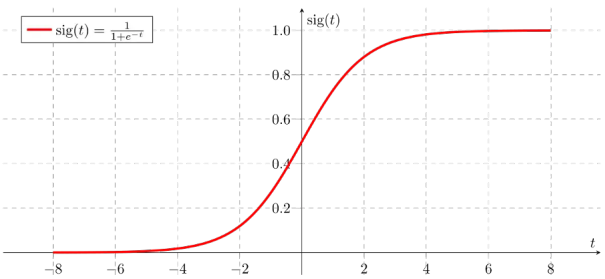
- Il bias misura quanto le previsioni del modello differiscono, in media, dal valore reale. Un alto bias generalmente indica che il modello è troppo semplice, non riuscendo a catturare la struttura sottostante dei dati, risultando in previsioni sistematicamente errate. Questo fenomeno è noto anche come 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*:

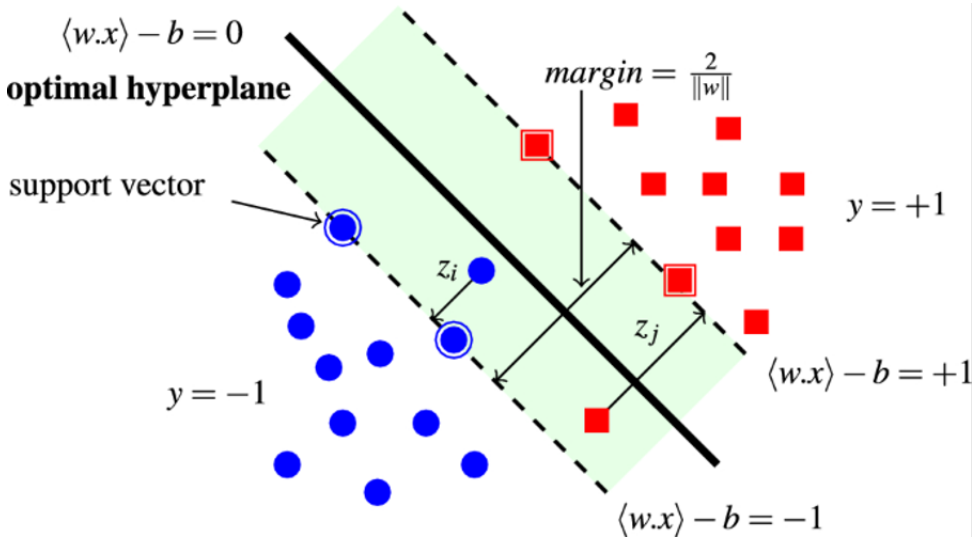
$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1-y_i)\log(1-p_i)]$$

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:

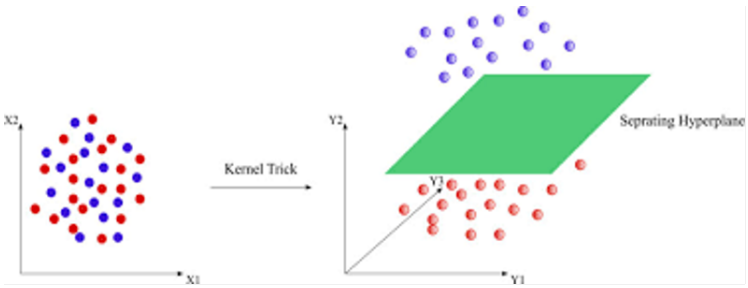
$$\min_{\vec{w},b,\xi} \frac{1}{2}\|\vec{w}\|^2 + C\sum_i \xi_i$$

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):

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

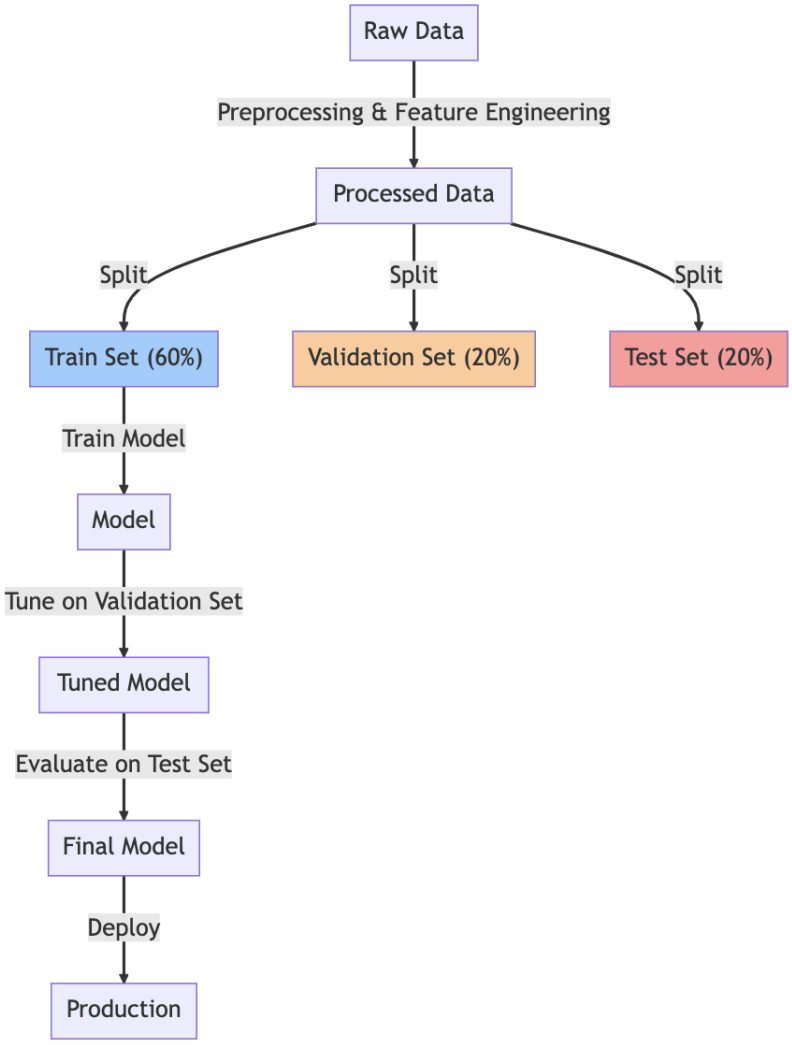


Cosa sono train validation e test

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.

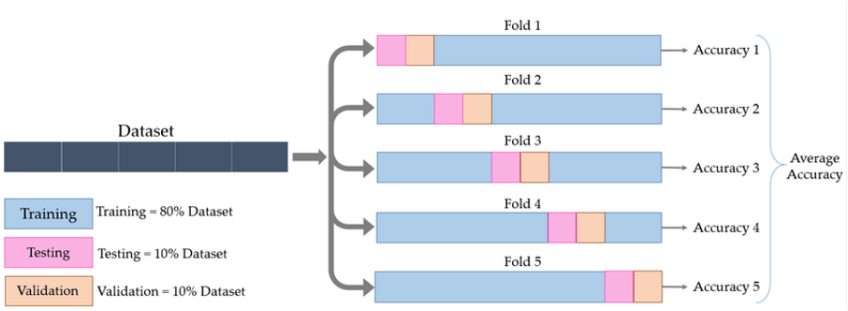
Una suddivisione comune dei dati è 60% train, 20% validation e 20% test. Tuttavia, è possibile utilizzare tecniche come la k-fold cross-validation per ottimizzare l'utilizzo dei dati disponibili, permettendo al modello di apprendere da diverse combinazioni dei dati.



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

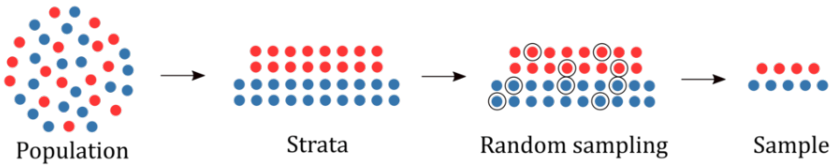
volta come set di dati di test. Alla fine, i risultati delle prove verranno combinati per ottenere una valutazione più precisa della performance del modello tramite la media.



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.