

# K-Nearest Neighbors

Lorenzo Arcioni

30 luglio 2024

## Sommario

In questo articolo, presentiamo un'analisi approfondita dell'algoritmo K-Nearest Neighbors (KNN), esaminandolo sia dal punto di vista teorico che pratico. L'algoritmo KNN è un metodo di apprendimento supervisionato utilizzato per la classificazione e la regressione, basato sul principio che oggetti simili sono vicini nello spazio delle caratteristiche. Iniziamo con una descrizione dettagliata dei fondamenti teorici del KNN, compresa la definizione formale, i criteri di scelta del parametro K e le metriche di distanza utilizzate per determinare la vicinanza tra i dati. Successivamente, esploriamo le sue proprietà matematiche e discutiamo l'impatto della dimensionalità dei dati e del rumore sulla sua performance. Attraverso un'analisi empirica, confrontiamo l'efficacia del KNN con altri algoritmi di machine learning, utilizzando dataset standard. Infine, esaminiamo le tecniche di ottimizzazione e miglioramento del KNN, come la normalizzazione dei dati e l'uso di pesi nei vicini, per aumentare la precisione e l'efficienza computazionale. Questo studio offre una visione completa del KNN, evidenziando i suoi punti di forza, le sue limitazioni e le situazioni in cui è più adatto.

## Indice

|          |   |          |
|----------|---|----------|
| <b>1</b> | <b>Introduzione</b>                                   | <b>1</b> |
| 1.1      | Panoramica dell'algoritmo K-Nearest Neighbors (KNN)   | 1        |
| 1.2      | Funzionamento di KNN                                  | 1        |
| 1.3      | Potenzialità di KNN                                   | 1        |
| 1.4      | Caratteristiche e Limitazioni                         | 2        |
| 1.5      | Definizione e concetto di base                        | 2        |
| 1.5.1    | Dataset, feature e variabile target                   | 2        |
| 1.5.2    | Problema di classificazione                           | 2        |
| 1.5.3    | Problema di regressione                               | 3        |
| 1.6      | Obiettivi dell'articolo                               | 3        |
| <b>2</b> | <b>Fondamenti Teorici del KNN</b>                     | <b>4</b> |
| 2.1      | Definizione formale                                   | 4        |
| 2.2      | Metriche di distanza                                  | 4        |
| 2.2.1    | Distanza Euclidea                                     | 5        |
| 2.2.2    | Distanza Manhattan                                    | 5        |
| 2.2.3    | Distanza Chebyshev                                    | 5        |
| 2.2.4    | Distanza Minkowski                                    | 6        |
| 2.2.5    | Distanza Coseno                                       | 6        |
| 2.2.6    | Distanza di Hamming                                   | 7        |
| 2.2.7    | Distanza Mahalanobis                                  | 7        |
| 2.3      | Algoritmo KNN   | 8        |
| <b>3</b> | <b>Analisi Teorica</b>                                | <b>8</b> |
| 3.1      | Interpretazione probabilistica                        | 8        |
| 3.1.1    | Convergenza   | 9        |
| 3.2      | La maledizione della dimensionalità                   | 9        |
| 3.2.1    | Concetto di Maledizione della Dimensionalità          | 9        |
| 3.2.2    | Impatto della Maledizione della Dimensionalità su KNN | 9        |

|       |  |           |
|-------|--|-----------|
| 3.2.3 | Mitigazione della Maledizione della Dimensionalità | 9         |
| 3.3   | Complessità computazionale                         | 10        |
| 3.3.1 | Costo Computazionale durante l'Addestramento       | 10        |
| 3.3.2 | Costo Computazionale durante la Predizione         | 10        |
| 3.3.3 | Effetti della Maledizione della Dimensionalità     | 10        |
| 3.3.4 | Ottimizzazione e Tecniche di Accelerazione         | 10        |
| 3.4   | Il Trade-Off Bias-Variance nel K-Nearest Neighbors | 11        |
| 3.4.1 | Errore di Bias e Errore di Varianza                | 11        |
| 3.4.2 | Trade-Off Bias-Variance in K-Nearest Neighbors     | 11        |
| 3.4.3 | Selezione del Valore Ottimale di $K$               | 12        |
| 4     | <b>Ottimizzazioni</b>                              | <b>12</b> |
| 4.1   | KD-Tree  | 12        |
| 4.2   | Ball Tree  | 12        |
| 4.3   | Hashing  | 12        |
| 4.4   | Algoritmi Approximate Nearest Neighbors (ANN)      | 12        |

# 1 Introduzione

## 1.1 Panoramica dell'algoritmo K-Nearest Neighbors (KNN)

L'algoritmo K-Nearest Neighbors (KNN) rappresenta un pilastro fondamentale nell'ambito dell'apprendimento automatico supervisionato, apprezzato per la sua semplicità concettuale e la sua efficacia in una vasta gamma di applicazioni. La sua filosofia si basa sul principio intuitivo che oggetti simili tendono a raggrupparsi nello stesso spazio delle caratteristiche. Questo approccio non parametrico permette a KNN di adattarsi a strutture dati complesse e a relazioni non lineari, senza fare assunzioni rigide sulla distribuzione dei dati.

## 1.2 Funzionamento di KNN

KNN opera determinando le etichette di classificazione o i valori di regressione per un nuovo punto, basandosi sulla vicinanza ai punti di addestramento. Il parametro chiave di KNN è  $K$ , che rappresenta il numero di "vicini" più prossimi da considerare durante la fase di predizione. Quando un nuovo dato deve essere classificato o valutato, KNN calcola la distanza (come vedremo, esistono varie tipologie di distanze) tra il dato da classificare, o valutare, e tutti i punti di addestramento; quindi seleziona i  $K$  punti più vicini. La classe o il valore di regressione del nuovo dato è determinato dalla classe maggiormente rappresentata o dalla media dei valori nei punti vicini, rispettivamente.

## 1.3 Potenzialità di KNN

KNN trova applicazione in numerosi settori grazie alla sua flessibilità e facilità di implementazione. Nei sistemi di raccomandazione, ad esempio, può suggerire prodotti o contenuti simili a quelli preferiti dall'utente sulla base dei gusti di altri utenti simili (vicini). Nell'analisi di immagini e nel riconoscimento di pattern, KNN può classificare nuove immagini confrontandole con esempi già noti. In ambito medico, può supportare la diagnosi confrontando i sintomi del paziente con casi storici simili.

Oltre ai settori menzionati, KNN può essere utilizzato in molti altri contesti. Nell'analisi di frodi, per esempio, KNN può identificare transazioni sospette confrontandole con transazioni conosciute come fraudolente. Nel settore del marketing, può segmentare i clienti in gruppi simili per creare campagne pubblicitarie mirate. Anche in ambito finanziario, KNN può essere utilizzato per prevedere l'andamento di titoli azionari o per valutare il rischio di credito basandosi su dati storici.

## 1.4 Caratteristiche e Limitazioni

Il KNN è noto per la sua semplicità e il principio della distanza su cui basare le decisioni di classificazione o regressione. Tuttavia, nella pratica, il KNN presenta diverse sfide significative.

Innanzitutto, KNN utilizza tutte le features in modo uguale, anche se alcune possono essere molto più predittive del target rispetto ad altre. Le distanze tra i punti vengono calcolate considerando

tutte le features in modo uguale, utilizzando metriche come la distanza Euclidea o Manhattan. Questo approccio, sebbene semplice, non è sempre il più efficace, poiché molte features possono essere irrilevanti per il target. Anche dopo aver effettuato una selezione delle features, la rilevanza delle stesse non sarà comunque uniforme.

Inoltre, le previsioni effettuate dai modelli KNN sono difficili da interpretare. Sebbene l'algoritmo sia comprensibile, capire le previsioni può essere complicato. È possibile elencare i  $K$  vicini più prossimi per un record, il che fornisce alcune indicazioni sulla previsione fatta per quel record, ma è difficile capire perché un determinato insieme di record sia considerato il più simile, soprattutto quando ci sono molte features in gioco.

Il KNN è anche sensibile al rumore nei dati e alla presenza di features non rilevanti, che possono influenzare negativamente le previsioni. Gestire grandi dataset con KNN può risultare computazionalmente oneroso, poiché richiede il calcolo delle distanze tra il nuovo dato e tutti i punti di addestramento. Questo problema diventa ancora più complesso quando si tratta di dataset con molte dimensioni (features).

Infine, quando i dati utilizzati da KNN sono altamente non strutturati, tipicamente non sono utili per comprendere la natura della relazione tra le features e il risultato della classe o della regressione. Questo limita ulteriormente l'interpretabilità del modello e la sua efficacia in situazioni reali.

## 1.5 Definizione e concetto di base

### 1.5.1 Dataset, feature e variabile target

Un dataset in Machine Learning è una collezione di dati organizzati in un formato strutturato. Ogni riga del dataset rappresenta un'osservazione, mentre ogni colonna rappresenta una caratteristica (feature) o la variabile target (etichetta). Le feature sono attributi che descrivono le osservazioni e possono essere di diversi tipi, come numeriche, categoriche o binarie. La variabile target è ciò che vogliamo predire utilizzando le feature.

Ad esempio, in un dataset per la predizione del prezzo delle case, le feature potrebbero includere la superficie della casa, il numero di camere, la posizione, l'anno di costruzione, ecc. La variabile target sarebbe il prezzo della casa.

### 1.5.2 Problema di classificazione

Consideriamo un esempio reale: la diagnosi precoce di malattie cardiache. Questo è un problema non banale che richiede l'uso del Machine Learning per essere risolto efficacemente. Il dataset potrebbe includere pazienti con varie caratteristiche cliniche misurate durante esami medici. Le feature potrebbero includere età, genere, pressione sanguigna, livelli di colesterolo, frequenza cardiaca massima, risultati di elettrocardiogrammi, e altre misure cliniche rilevanti. La variabile target sarebbe una variabile binaria che indica la presenza o l'assenza di una malattia cardiaca.

L'algoritmo KNN può essere utilizzato per classificare un nuovo paziente come "a rischio" o "non a rischio" di malattia cardiaca basandosi sui dati storici di altri pazienti. Quando un nuovo paziente entra per una valutazione, KNN calcola le distanze tra le caratteristiche cliniche del nuovo paziente e quelle dei pazienti nel dataset di addestramento. Seleziona i  $K$  pazienti più simili (i vicini più prossimi) e determina la classe del nuovo paziente in base alla maggioranza delle classi dei vicini.

Per esempio, se  $K = 5$  e tra i 5 pazienti più vicini al nuovo paziente 3 hanno una malattia cardiaca e 2 no, KNN predice che il nuovo paziente è "a rischio" di malattia cardiaca. Questa previsione può aiutare i medici a prendere decisioni informate riguardo ulteriori test o trattamenti, dimostrando l'importanza e l'utilità del Machine Learning in contesti medici critici.

### 1.5.3 Problema di regressione

Un esempio di problema di regressione risolvibile con l'algoritmo KNN è la previsione del valore di mercato delle proprietà immobiliari in una città. Questo problema richiede un approccio che si affida al Machine Learning per ottenere stime accurate e affidabili, data la moltitudine di fattori che influenzano i prezzi delle case.

Consideriamo un dataset che include informazioni dettagliate sulle proprietà immobiliari di una città. Le feature possono includere:

- Superficie della proprietà (in metri quadrati)
- Numero di camere da letto
- Numero di bagni
- Anno di costruzione
- Distanza dai servizi principali (scuole, ospedali, trasporti pubblici)
- Valutazioni della qualità del quartiere
- Prezzi recenti delle proprietà vicine

La variabile target in questo caso è il prezzo di vendita della proprietà.

Quando si vuole stimare il valore di una nuova proprietà, l'algoritmo KNN calcola la distanza tra le caratteristiche della nuova proprietà e quelle delle proprietà nel dataset di addestramento. Utilizzando una metrica di distanza (come la distanza euclidea), KNN identifica i  $K$  immobili più simili.

Ad esempio, se  $K = 5$ , KNN selezionerà le cinque proprietà più vicine alla nuova proprietà in termini di caratteristiche. Il prezzo stimato per la nuova proprietà sarà la media dei prezzi delle cinque proprietà più vicine. Il prezzo stimato della nuova proprietà sarà dunque

$$\hat{y} = \frac{1}{K} \sum_{i \in \mathcal{N}_K(\mathbf{x})} y_i$$

Dove  $K$  è il numero di vicini considerati,  $\mathcal{N}_K(\mathbf{x})$  rappresenta l'insieme dei  $K$  vicini più prossimi e  $y_i$  è il prezzo della proprietà  $i$ -esima.

Questo approccio di regressione basato su KNN è particolarmente utile perché tiene conto della località spaziale e delle caratteristiche specifiche delle proprietà immobiliari. Inoltre, permette di adattarsi a variazioni non lineari e complesse nei dati, che sono comuni nel mercato immobiliare. La previsione accurata dei prezzi immobiliari è fondamentale per acquirenti, venditori, agenti immobiliari e investitori, rendendo KNN uno strumento prezioso in questo contesto.

## 1.6 Obiettivi dell'articolo

L'obiettivo principale di questo articolo è fornire una comprensione completa e dettagliata dell'algoritmo K-Nearest Neighbors (KNN) attraverso un'analisi sia teorica che pratica.

In primo luogo, l'articolo presenta i fondamenti teorici di KNN, spiegando il funzionamento dell'algoritmo, le metriche di distanza utilizzate (come la distanza euclidea e la distanza di Manhattan) e l'importanza della scelta del parametro  $K$ . Verranno, inoltre, discusse le implicazioni di queste scelte sulla performance dell'algoritmo.

In secondo luogo, verranno esaminate le proprietà matematiche di KNN, inclusa la complessità computazionale e le sfide legate alla "maledizione della dimensionalità". Saranno analizzati i trade-off tra bias e varianza per comprendere come ottimizzare le prestazioni di questo algoritmo.

In terzo luogo, l'articolo fornirà un'analisi empirica, confrontando KNN con altri algoritmi di Machine Learning su dataset standard. Questo confronto aiuterà a evidenziare i punti di forza e le limitazioni di KNN in scenari pratici.

Infine, l'articolo esplora le tecniche di ottimizzazione e miglioramento di KNN, come la normalizzazione dei dati, l'uso di KNN pesato e l'implementazione di strutture dati efficienti come KD-Trees.

## 2 Fondamenti Teorici del KNN

### 2.1 Definizione formale

Per formalizzare matematicamente l'algoritmo K-Nearest Neighbors (KNN), consideriamo un dataset di addestramento  $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ , dove  $\mathbf{x}_i \in \mathbb{R}^d$  rappresenta un vettore di lunghezza  $d$  (un punto dati con  $d$  caratteristiche) e  $y_i \in \mathbb{R}$  (o  $y_i \in \{1, \dots, C\}$  dove  $C \in \mathbb{N}$ , per la classificazione) rappresenta un'istanza della variabile target associata.

Assumiamo di aver osservato un insieme di  $n$  punti dati diversi. Queste osservazioni sono chiamate dati di addestramento perché li utilizziamo per addestrare il nostro modello su come stimare una funzione che ci permette di formalizzare la realtà di interesse. Lasciamo che  $x_{ij}$  rappresenti il valore del  $j$ -esimo predittore, o input, per l'osservazione  $i$ , dove  $i = 1, 2, \dots, n$  e  $j = 1, 2, \dots, d$ . Di conseguenza, sia  $y_i$  la variabile di risposta per l'osservazione  $i$ . I nostri dati di addestramento consistono in  $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ , dove  $x_i = (x_{i1}, x_{i2}, \dots, x_{id})^T$ .

Definiamo ora due funzioni utili a modellare il problema statistico:

- La funzione reale  $f$  descrive la relazione tra l'input  $\mathbf{x}$  e la variabile target  $y$ . Questa funzione rappresenta il fenomeno o il processo che genera i dati osservati.
- La funzione stimata  $\hat{f}$  è l'approssimazione di  $f$  ottenuta attraverso un modello statistico o un algoritmo di apprendimento (KNN in questo caso).  $\hat{f}$  cerca di approssimare  $f$  utilizzando i dati di addestramento disponibili per fare previsioni o inferenze su nuovi dati non osservati.

Il nostro obiettivo è applicare un metodo di apprendimento statistico ai dati di addestramento per stimare la funzione sconosciuta  $f$ . In altre parole, vogliamo trovare una funzione  $\hat{f}$  tale che  $y \approx \hat{f}(\mathbf{x})$  per qualsiasi osservazione  $(\mathbf{x}, y)$ .

I metodi non parametrici, come il KNN, non presuppongono una forma specifica per la funzione  $f$ . Invece, cercano di stimare  $f$  in modo che si avvicini il più possibile ai dati osservati, mantenendo un andamento fluido e coerente. Questi approcci offrono un vantaggio significativo rispetto ai metodi parametrici poiché non limitano  $f$  a una forma predeterminata, permettendo quindi di adattarsi meglio a una vasta gamma di possibili configurazioni per  $f$ . Nei metodi parametrici, invece, c'è il rischio che la forma funzionale  $\hat{f}$ , utilizzata per stimare  $f$ , sia molto diversa dalla reale  $f$ , portando a modelli che non si adattano bene ai dati. Al contrario, i metodi non parametrici evitano completamente questo rischio perché non impongono alcuna assunzione sulla forma di  $f$ . Tuttavia, gli approcci non parametrici hanno uno svantaggio principale: essi richiedono un numero elevato di osservazioni per ottenere una stima accurata di  $f$ , molto più alto rispetto a quanto necessario nei metodi parametrici che riducono il problema a un numero limitato di parametri.

## 2.2 Metriche di distanza

Per determinare i  $K$  vicini più prossimi, è necessario definire una metrica di distanza  $d(\mathbf{x}, \mathbf{z})$  tra due vettori (punti dati)  $\mathbf{x}$  e  $\mathbf{z}$ . Prima di parlare di metriche di distanza, è doveroso fare un'introduzione riguardo alla norma.

Una **norma** è una funzione che associa a ciascun vettore di uno spazio vettoriale un numero reale non negativo e soddisfa specifiche proprietà di compatibilità con la struttura dello spazio vettoriale. L'obiettivo principale di una norma è fornire una nozione di "lunghezza" dei vettori nello spazio vettoriale considerato. Le proprietà di compatibilità con la struttura di spazio vettoriale sono pensate per riflettere alcune caratteristiche intuitive della "lunghezza" quando si effettuano operazioni come l'addizione di vettori o la moltiplicazione di un vettore per uno scalare.

Una norma su uno spazio vettoriale reale o complesso  $\mathcal{X}$ , è una funzione:

$$\begin{aligned} \|\cdot\|: \mathcal{X} &\rightarrow \mathbb{R} \\ \mathbf{x} &\mapsto \|\mathbf{x}\| \end{aligned}$$

che verifica le seguenti condizioni:

- $\|\mathbf{x}\| \geq 0$ , per ogni  $\mathbf{x} \in \mathcal{X}$ ;
- $\|\mathbf{x}\| = 0$  se e solo se  $\mathbf{x} = \mathbf{0}$ ;
- $\|\lambda\mathbf{x}\| = |\lambda|\|\mathbf{x}\|$ , per ogni scalare  $\lambda$  (omogeneità);
- $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ , per ogni  $\mathbf{x}, \mathbf{y} \in \mathcal{X}$  (disuguaglianza triangolare).

La coppia  $(\mathcal{X}, \|\cdot\|)$  costituisce uno spazio normato.

Una metrica di distanza è una funzione che misura la distanza tra due punti in uno spazio. Formalmente, una funzione  $d(\mathbf{x}, \mathbf{z})$  è una metrica se soddisfa le seguenti proprietà:

- **Non Negatività:**  $d(\mathbf{x}, \mathbf{z}) \geq 0$  per ogni coppia di punti  $\mathbf{x}$  e  $\mathbf{z}$ .
- **Definizione Positiva:**  $d(\mathbf{x}, \mathbf{z}) = 0$  se e solo se  $\mathbf{x} = \mathbf{z}$ .
- **Simmetria:**  $d(\mathbf{x}, \mathbf{z}) = d(\mathbf{z}, \mathbf{x})$  per ogni coppia di punti  $\mathbf{x}$  e  $\mathbf{z}$ .
- **Disuguaglianza Triangolare:**  $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$  per ogni trio di punti  $\mathbf{x}$ ,  $\mathbf{y}$ , e  $\mathbf{z}$ .

### 2.2.1 Distanza Euclidea

La distanza Euclidea (chiamata anche norma  $L_2$ ) calcola la distanza diretta tra punti in uno spazio multidimensionale. È adatta per dati che presentano caratteristiche continue e numeriche con scale e intervalli simili. Può anche gestire bene gli outlier e il rumore, poiché dà più peso alle differenze più grandi. Tuttavia, può essere influenzata dal "curse of dimensionality", che significa che all'aumentare del numero di features, la distanza tra due punti diventa meno significativa e più simile tra loro. Inoltre, può essere influenzata dall'orientamento e dalla scala delle caratteristiche, poiché assume che tutte le features siano ugualmente importanti e che abbiano tutte una stessa scala.

$$d(\mathbf{x}, \mathbf{z}) = \|\mathbf{x} - \mathbf{z}\| = \sqrt{\sum_{j=1}^d (x_j - z_j)^2}$$

### 2.2.2 Distanza Manhattan

La distanza Manhattan (chiamata anche norma  $L_1$ ) è una metrica adatta per dati che presentano caratteristiche discrete e categoriali, poiché non penalizza tanto le piccole differenze quanto la distanza euclidea. Inoltre, gestisce meglio i dati ad alta dimensionalità, poiché è meno sensibile al "curse of dimensionality". Tuttavia, può essere influenzata dall'orientamento e dalla scala delle caratteristiche, poiché assume che tutte le features siano ugualmente importanti e che abbiano tutte una stessa scala.

$$d(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^d |x_j - z_j|$$

### 2.2.3 Distanza Chebyshev

La distanza Chebyshev (chiamata anche norma  $L_\infty$ ) è adatta per dati che presentano caratteristiche discrete e categoriali, poiché non penalizza tanto le piccole differenze quanto la distanza euclidea. Inoltre, gestisce meglio i dati ad alta dimensionalità, poiché è meno sensibile al "curse of dimensionality". Tuttavia, è influenzata dall'orientamento e dalla scala delle caratteristiche, poiché assume che tutte le features siano ugualmente importanti e che abbiano tutte una stessa scala.

$$d(\mathbf{x}, \mathbf{z}) = \sqrt[\infty]{\sum_{j=1}^d |x_j - z_j|^\infty} = \max_{j=1}^d |x_j - z_j|$$

Diamo ora una dimostrazione della seconda uguaglianza. Siano  $\mathbf{x}$  e  $\mathbf{z}$  due vettori di dimensione  $d$ .

Sia  $\mathbf{a} = [|x_i - z_i|]_{i=1}^d$  il vettore differenza in valore assoluto tra  $\mathbf{x}$  e  $\mathbf{z}$ .

Senza perdita di generalità, supponiamo che  $\max_{j=1}^d |x_j - z_j| = a_i$ .

Allora:

$$\lim_{p \rightarrow \infty} (a_1^p + \dots + a_d^p)^{1/p} \geq \lim_{p \rightarrow \infty} (a_i^p)^{1/p} = \lim_{p \rightarrow \infty} a_i = a_i = \max_{j=1}^d |x_j - z_j|$$

e:

$$\begin{aligned}
\lim_{p \rightarrow \infty} (a_1^p + \dots + a_d^p)^{1/p} &\leq \lim_{p \rightarrow \infty} (a_i^p + \dots + a_i^p)^{1/p} = \lim_{p \rightarrow \infty} (d \cdot a_i^p)^{1/p} \\
&= \lim_{p \rightarrow \infty} a_i \cdot d^{1/p} \\
&= a_i \cdot \lim_{p \rightarrow \infty} d^{1/p} \\
&= a_i = \max_{j=1}^d |x_j - z_j|
\end{aligned}$$

Quindi, per il teorema del confronto,  $\lim_{p \rightarrow \infty} (a_1^p + \dots + a_d^p)^{1/p} = \max_{j=1}^d |x_j - z_j|$ .  $\square$

## 2.2.4 Distanza Minkowski

La distanza Minkowski (chiamata anche norma  $L_p$ ) è una generalizzazione delle distanze Euclidea, Manhattan e Chebyshev. È definita da un parametro  $p$  che controlla quanto peso viene dato alle differenze più grandi o più piccole tra le coordinate.

$$d(\mathbf{x}, \mathbf{z}) = \left( \sum_{j=1}^d |x_j - z_j|^p \right)^{\frac{1}{p}}$$

dove  $p$  è un parametro positivo che determina la forma della distanza. La distanza Minkowski può essere vista come una famiglia di metriche di distanza che include la distanza Euclidea ( $p = 2$ ), la distanza di Manhattan ( $p = 1$ ) e la distanza di Chebyshev ( $p = \infty$ ), che rappresenta il massimo delle differenze assolute tra le coordinate. La distanza di Minkowski è adatta per dati che presentano tipi misti di caratteristiche, poiché consente di regolare il parametro  $p$  per bilanciare l'importanza delle diverse caratteristiche e delle distanze. Tuttavia, può essere computazionalmente costosa e difficile da interpretare, poiché il parametro  $p$  può avere effetti diversi su diversi insiemi di dati e problemi.

## 2.2.5 Distanza Coseno

La distanza coseno (o similarità coseno) misura l'angolo tra due vettori in uno spazio multidimensionale, piuttosto che la loro distanza diretta. È particolarmente utile per dati in cui l'orientamento dei vettori è più importante della loro magnitudine, come nel caso di dati testuali o vettori di parole (word embeddings). La distanza coseno è calcolata come uno meno il coseno dell'angolo tra i vettori, quindi varia tra 0 e 2, dove 0 indica vettori identici (completamente simili) e 2 indica vettori diametralmente opposti (completamente dissimili).

$$d(\mathbf{x}, \mathbf{z}) = 1 - \frac{\mathbf{x} \cdot \mathbf{z}}{\|\mathbf{x}\| \cdot \|\mathbf{z}\|} = 1 - \frac{\sum_{j=1}^d x_j z_j}{\sqrt{\sum_{j=1}^d x_j^2} \sqrt{\sum_{j=1}^d z_j^2}}$$

Qui,  $\mathbf{x} \cdot \mathbf{z}$  rappresenta il prodotto scalare tra i vettori  $\mathbf{x}$  e  $\mathbf{z}$ , mentre  $\|\mathbf{x}\|$  e  $\|\mathbf{z}\|$  sono le norme euclidee dei rispettivi vettori.

La distanza coseno è particolarmente robusta rispetto a variazioni di scala nei dati, poiché normalizza i vettori prima di calcolare l'angolo tra essi. Questo la rende adatta per scenari in cui la lunghezza assoluta (magnitudine) dei vettori è meno rilevante rispetto alla direzione, come nell'analisi di documenti o nella raccomandazione di oggetti basata su preferenze utente. Tuttavia, può essere influenzata dalla presenza di valori nulli o zero nei dati, che possono distorcere il calcolo della similarità.

Diamo ora un'interpretazione intuitiva della distanza coseno. Siano  $\mathbf{x}$  e  $\mathbf{z}$  due vettori di dimensione  $d$ . Il coseno dell'angolo  $\alpha$  (l'angolo con ampiezza minore tra i due vettori), tra  $\mathbf{x}$  e  $\mathbf{z}$  si ottiene, secondo la definizione di coseno, dividendo  $\|proj_{\mathbf{z}} \mathbf{x}\|$  (la lunghezza della proiezione del vettore  $\mathbf{x}$  nella direzione del vettore  $\mathbf{z}$ ) per la norma del vettore  $\mathbf{x}$ :

$$\cos(\alpha) = \frac{\|proj_{\mathbf{z}} \mathbf{x}\|}{\|\mathbf{x}\|}. \quad (1)$$

Ora non ci rimane che trovare la norma del vettore proiezione. La proiezione  $proj_{\mathbf{z}} \mathbf{x}$  è un vettore che giace nella direzione di  $\mathbf{z}$ , quindi possiamo scriverlo come  $c \cdot \mathbf{z}$  per qualche numero  $c$ . Per trovare

$c$  possiamo utilizzare il fatto che la proiezione  $proj_z \mathbf{x} = c \cdot \mathbf{z}$  è un vettore che si annulla quando  $\mathbf{x}$  è perpendicolare a  $\mathbf{z}$ . Come vettore perpendicolare a  $\mathbf{z}$  possiamo considerare il vettore  $\mathbf{x} - c \cdot \mathbf{z}$  e moltiplicarlo per  $\mathbf{z}$ . Possiamo quindi ricavare il valore  $c$  dalla seguente equazione:

$$(\mathbf{x} - c \cdot \mathbf{z}) \cdot \mathbf{z} = 0 \quad (2)$$

$$\mathbf{x} \cdot \mathbf{z} - c \cdot \mathbf{z} \cdot \mathbf{z} = 0 \quad (3)$$

$$\mathbf{x} \cdot \mathbf{z} - c \|\mathbf{z}\|^2 = 0 \quad (4)$$

$$c = \frac{\mathbf{x} \cdot \mathbf{z}}{\|\mathbf{z}\|^2}. \quad (5)$$

Ora, mettendo insieme le due equazioni (1) e (5), otteniamo

$$\cos(\alpha) = \frac{\|proj_z \mathbf{x}\|}{\|\mathbf{x}\|} = \frac{\|c \cdot \mathbf{z}\|}{\|\mathbf{x}\|} = \frac{c \cdot \|\mathbf{z}\|}{\|\mathbf{x}\|} = \frac{\mathbf{x} \cdot \mathbf{z} \|\mathbf{z}\|}{\|\mathbf{z}\|^2 \|\mathbf{x}\|} = \frac{\mathbf{x} \cdot \mathbf{z}}{\|\mathbf{x}\| \cdot \|\mathbf{z}\|}.$$

Per ottenere ora un valore positivo che indica la distanza coseno tra due vettori, sottraiamo il coseno, che ha valori in  $[-1, 1]$ , ad 1. In questo modo otteniamo una distanza massima di 2 e una distanza minima di 0.  $\square$

### 2.2.6 Distanza di Hamming

La distanza di Hamming è una metrica comunemente utilizzata per misurare la somiglianza o la dissimiglianza tra due vettori di caratteristiche binarie. È particolarmente utile quando si lavora con dati categorici o binari, dove ogni caratteristica può assumere solo due valori possibili.

Consideriamo due vettori di caratteristiche,  $\mathbf{x}$  e  $\mathbf{z}$ , ciascuno composto da  $d$  caratteristiche binarie.

La distanza di Hamming tra  $\mathbf{x}$  e  $\mathbf{z}$ , denotata come  $d(\mathbf{x}, \mathbf{z})$ , può essere calcolata utilizzando la seguente formula:

$$d(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^d (x_i \oplus z_i)$$

dove  $\oplus$  denota l'operazione XOR elemento per elemento. Questo significa che la distanza di Hamming è la somma delle differenze bit a bit tra le caratteristiche corrispondenti dei due vettori.

Per chiarire, l'operazione XOR ( $\oplus$ ) restituisce 1 se  $x_i \neq z_i$  e 0 se sono uguali. Pertanto, sommando tutti i risultati degli XOR si ottiene il conteggio delle posizioni in cui i due vettori differiscono.

Una variante generalizzata, utilizzabile per vettori di features categoriche (non per forza binarie), della distanza di Hamming può essere calcolata utilizzando la seguente formula:

$$d(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^d \mathbf{1}_{x_i \neq z_i}$$

dove  $x_i \neq z_i$  indica che la caratteristica  $i$  del vettore  $\mathbf{x}$  è diversa da quella del vettore  $\mathbf{z}$ .

La distanza di Hamming è particolarmente conveniente da utilizzare quando le caratteristiche dei dati sono binarie (derivate ad esempio da un one-hot encoding) o categoriche che possono essere codificate in forma numerica.

### 2.2.7 Distanza Mahalanobis

[...]

## 2.3 Algoritmo KNN

Nel KNN, la variabile target  $\hat{y}$  di un nuovo punto dati  $\hat{\mathbf{x}}$  viene determinata come segue:



1. Calcolare la distanza, utilizzando la metrica di distanza scelta, tra  $\hat{\mathbf{x}}$  e ogni punto dati  $\mathbf{x}_i$  nel dataset di addestramento  $\mathcal{D}$ . Definiamo un nuovo vettore  $\mathbf{d}^{\hat{\mathbf{x}}}$  che contiene le distanze tra  $\hat{\mathbf{x}}$  e ogni punto di  $\mathcal{D}$ :

$$\mathbf{d}^{\hat{\mathbf{x}}} = [d(\hat{\mathbf{x}}, \mathbf{x}_i)]_{i=1}^N$$

2. A partire da  $\mathbf{d}^{\hat{\mathbf{x}}} = [d_1^{\hat{\mathbf{x}}}, \dots, d_N^{\hat{\mathbf{x}}}]$ , determiniamo un insieme  $\mathcal{N}_K(\hat{\mathbf{x}}) = \{i_1, \dots, i_k\}$  contenente gli indici dei  $K$  vicini più prossimi di  $\hat{\mathbf{x}}$ , quindi tale che

$$\forall i \in \mathcal{N}_K(\hat{\mathbf{x}}) \quad \nexists j \in [N] \setminus \mathcal{N}_K(\hat{\mathbf{x}}) : d_j^{\hat{\mathbf{x}}} < d_i^{\hat{\mathbf{x}}}.$$

3. Assegnare a  $\hat{y}$ :

- **Classificazione:** la classe più frequente tra i  $K$  vicini più prossimi. Formalmente,

$$\hat{y} = \arg \max_{c \in \{1, \dots, C\}} \sum_{i \in \mathcal{N}_K(\hat{\mathbf{x}})} \mathbf{1}_{\{y_i=c\}}$$

dove  $\mathcal{N}_K(\hat{\mathbf{x}})$  denota l'insieme degli indici dei  $K$  vicini più prossimi di  $\hat{\mathbf{x}}$  e  $\mathbf{1}_{\{y_i=c\}}$  è una funzione indicatrice che vale 1 se  $y_i = c$  e 0 altrimenti.

La funzione di aggregazione più comunemente utilizzata è la moda (la classe più frequente), ma è possibile utilizzarne altre.

- **Regresione:** il risultato della funzione di aggregazione (solitamente la media aritmetica) applicata ai valori dei  $K$  vicini più prossimi. Formalmente,

$$\hat{y} = \frac{1}{K} \sum_{i \in \mathcal{N}_K(\hat{\mathbf{x}})} y_i$$

nel caso della media aritmetica. Per stabilire il valore da assegnare alla variabile target  $\hat{y}$ , possono essere utilizzate anche altre funzioni di aggregazione, come la mediana, la moda o la media ponderata in base alla distanza dei vicini (minore distanza dal punto = maggiore peso nel calcolo della media). In particolare, in quest'ultimo caso l'algoritmo prende il nome di Weighted KNN.

## 3 Analisi Teorica

### 3.1 Interpretazione probabilistica

In teoria, per effettuare previsioni accurate, sarebbe ideale conoscere la distribuzione condizionale dei dati. Tuttavia, nella pratica, questa distribuzione è generalmente sconosciuta, rendendo impossibile una stima diretta basata su di essa. Nonostante ciò, metodi come il K-nearest neighbors (KNN) riescono comunque a fare previsioni accurate stimando tale distribuzione in maniera non parametrica.

Il KNN stima la distribuzione dei dati basandosi sui  $K$  punti di addestramento più vicini a un punto di test  $\hat{\mathbf{x}}$ . La probabilità condizionale viene calcolata come la frazione dei punti in questo insieme che condividono la stessa caratteristica della variabile di interesse:

$$Pr(Y = j \mid X = \mathbf{x}_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j),$$

dove  $N_0$  rappresenta l'insieme dei  $K$  punti di addestramento più vicini a  $\mathbf{x}_0$  e  $I(y_i = j)$  è una funzione indicatrice che vale 1 se  $y_i$  è uguale a  $j$  e 0 altrimenti.

Nonostante la semplicità del metodo, il KNN può spesso produrre previsioni molto efficaci, avvicinandosi al comportamento ottimale in molti scenari. Tuttavia, la scelta del parametro  $K$  è cruciale: un valore troppo piccolo di  $K$  rende il modello troppo flessibile e sensibile al rumore nei dati, mentre un valore troppo grande può rendere il modello eccessivamente rigido e incapace di catturare la struttura sottostante dei dati.

La relazione tra il tasso di errore di addestramento e quello di test non è sempre diretta. Aumentando la flessibilità del modello (diminuendo  $K$ ), il tasso di errore di addestramento tende a diminuire, ma l'errore di test può aumentare se il modello soffre di overfitting. Questo comportamento è ben rappresentato dalla forma a U del grafico dell'errore di test in funzione di  $1/K$ .

La scelta del giusto livello di flessibilità è fondamentale per il successo di qualsiasi metodo di apprendimento statistico. Nel Capitolo 5, torneremo su questo argomento e discuteremo vari metodi per stimare i tassi di errore di test, al fine di scegliere il livello ottimale di flessibilità per un determinato metodo di apprendimento statistico.

### 3.1.1 Convergenza

## 3.2 La maledizione della dimensionalità

La *maledizione della dimensionalità* è un concetto fondamentale in statistica e machine learning che descrive come la qualità delle analisi e delle previsioni possa deteriorarsi con l'aumento del numero di dimensioni (o variabili) nel dataset. Questo fenomeno diventa particolarmente rilevante quando si utilizzano algoritmi di apprendimento automatico come K-Nearest Neighbors (KNN).

### 3.2.1 Concetto di Maledizione della Dimensionalità

Quando i dati sono rappresentati in spazi ad alta dimensionalità, le distanze tra i punti tendono a diventare sempre più simili, il che complica la capacità di un algoritmo di distinguere tra punti vicini e lontani. Questo comportamento può influenzare negativamente la performance di molti algoritmi, inclusi i metodi basati sulla distanza come KNN.

### 3.2.2 Impatto della Maledizione della Dimensionalità su KNN

Quando il numero di dimensioni  $d$  aumenta, il volume dello spazio aumenta esponenzialmente. Questo comporta che la distanza tra tutti i punti aumenta, e la differenza tra le distanze dei vicini diventa meno pronunciata. In altre parole, in uno spazio ad alta dimensionalità, quasi tutti i punti sembrano equidistanti l'uno dall'altro. Questo ha le seguenti implicazioni per KNN:

- **Distanza meno informativa:** Le distanze calcolate tra i punti diventano meno informative. In spazi ad alta dimensionalità, le distanze tra punti vicini e lontani tendono a convergere, rendendo difficile identificare i vicini più prossimi con precisione.
- **Sparsità dei dati:** I dati diventano sparsi in spazi ad alta dimensionalità. Questo significa che ogni punto di dati è relativamente lontano dagli altri punti, riducendo la densità del campione e aumentando l'incertezza nella classificazione o nella regressione.
- **Maggiore costo computazionale:** Con l'aumento delle dimensioni, il costo computazionale per calcolare le distanze tra tutti i punti aumenta, rendendo l'algoritmo meno scalabile e più lento.

### 3.2.3 Mitigazione della Maledizione della Dimensionalità

Per contrastare gli effetti negativi della maledizione della dimensionalità su KNN, è possibile adottare diverse strategie:

- **Riduzione della dimensionalità:** Tecniche come l'Analisi delle Componenti Principali (PCA) o il t-Distributed Stochastic Neighbor Embedding (t-SNE) possono essere utilizzate per ridurre il numero di dimensioni mantenendo il più possibile la struttura originale dei dati.
- **Selezione delle caratteristiche:** Identificare e mantenere solo le caratteristiche più rilevanti può aiutare a ridurre la dimensionalità e migliorare la performance di KNN.
- **Normazione dei dati:** Applicare tecniche di normazione o scalatura per uniformare le scale delle diverse dimensioni può aiutare a migliorare la coerenza delle distanze calcolate.

In conclusione, la maledizione della dimensionalità rappresenta una sfida significativa per algoritmi basati sulla distanza come KNN, ma l'adozione di tecniche adeguate di riduzione e selezione delle caratteristiche può mitigare questi effetti e migliorare la performance degli algoritmi in spazi ad alta dimensionalità.

### 3.3 Complessità computazionale

L'algoritmo K-Nearest Neighbors (KNN) è noto per la sua semplicità e facilità di implementazione, ma il suo costo computazionale può essere considerevole, specialmente con grandi volumi di dati e in spazi ad alta dimensionalità. In questa sottosezione, analizzeremo il costo computazionale di KNN sia durante la fase di addestramento che durante la fase di predizione.

#### 3.3.1 Costo Computazionale durante l'Addestramento

Uno dei punti di forza di KNN è che non richiede una fase di addestramento vera e propria. In altre parole, l'algoritmo non costruisce un modello durante la fase di addestramento; invece, memorizza semplicemente i dati di addestramento. Di conseguenza, il costo computazionale dell'addestramento è:

$$O(1) \tag{6}$$

Questo significa che il tempo richiesto per "addestrare" KNN è costante e non dipende dalla dimensione del dataset. Tuttavia, è importante notare che, anche se il costo di addestramento è trascurabile, le risorse di memoria devono essere sufficienti per memorizzare l'intero dataset di addestramento.

#### 3.3.2 Costo Computazionale durante la Predizione

Il costo computazionale durante la fase di predizione è significativamente più elevato rispetto alla fase di addestramento. Per ogni nuovo punto di test, KNN deve calcolare la distanza tra il punto di test e tutti i punti di addestramento per determinare i K vicini più prossimi. Questo può essere espresso come segue:

$$O(n \cdot d) \tag{7}$$

dove  $n$  è il numero di punti di addestramento e  $d$  è il numero di dimensioni. Questa complessità deriva dal fatto che l'algoritmo calcola la distanza Euclidea tra il punto di test e ogni punto di addestramento, e la distanza Euclidea ha una complessità di  $O(d)$  per ogni calcolo. Poiché questo calcolo deve essere effettuato per tutti i  $n$  punti, il costo totale è  $O(n \cdot d)$ .

#### 3.3.3 Effetti della Maledizione della Dimensionalità

La maledizione della dimensionalità amplifica ulteriormente il costo computazionale di KNN. Con l'aumento del numero di dimensioni  $d$ , il costo di calcolo delle distanze aumenta linearmente, e la distanza tra i punti diventa meno informativa. Questo porta a una maggiore difficoltà nel trovare i veri vicini più prossimi e, di conseguenza, a una maggiore richiesta di calcoli. In spazi ad alta dimensionalità, il costo computazionale può crescere in modo esponenziale con il numero di dimensioni.

#### 3.3.4 Ottimizzazione e Tecniche di Accelerazione

Per affrontare il problema del costo computazionale, esistono diverse tecniche di ottimizzazione e accelerazione che possono essere applicate:

- **Strutture di Dati di Indice:** L'uso di strutture di dati come gli alberi k-d, gli alberi di ricerca spaziale o i grafi di prossimità può ridurre significativamente il costo computazionale. Queste strutture permettono di eseguire query di vicinanza più velocemente rispetto a una ricerca esaustiva.

- **Approssimazione:** Algoritmi di ricerca approssimativa dei vicini più prossimi, come l'Approximate Nearest Neighbors (ANN), possono fornire risultati vicini a quelli esatti con un costo computazionale ridotto. Tecniche come Locality Sensitive Hashing (LSH) sono utilizzate per trovare vicini approssimativi in modo più efficiente.
- **Riduzione della Dimensionalità:** Tecniche di riduzione della dimensionalità, come PCA o t-SNE, possono essere utilizzate per ridurre il numero di dimensioni  $d$ , diminuendo così il costo di calcolo delle distanze e migliorando la performance dell'algoritmo.
- **Parallelizzazione:** L'algoritmo può essere parallelizzato per sfruttare più core di CPU o GPU, accelerando i calcoli delle distanze per punti di test multipli.

In sintesi, sebbene KNN sia un algoritmo semplice e non richieda un costoso processo di addestramento, la sua fase di predizione può diventare molto costosa in termini di tempo di calcolo, specialmente con grandi dataset e in spazi ad alta dimensionalità. L'adozione di tecniche di ottimizzazione e accelerazione può aiutare a gestire e ridurre il costo computazionale associato a questo algoritmo.

### 3.4 Il Trade-Off Bias-Variance nel K-Nearest Neighbors

Il trade-off bias-variance è un concetto cruciale nella valutazione e nel miglioramento delle prestazioni degli algoritmi di apprendimento automatico. Questo trade-off riguarda la relazione tra due tipi di errore che un modello può commettere: l'errore di bias e l'errore di varianza. Per l'algoritmo K-Nearest Neighbors (KNN), questo trade-off è particolarmente rilevante e si manifesta in modo distintivo a seconda del valore di  $K$ , il numero di vicini utilizzati per la predizione.

#### 3.4.1 Errore di Bias e Errore di Varianza

Prima di esaminare il trade-off nel contesto di KNN, è utile definire i concetti di bias e varianza:

- **Errore di Bias:** Il bias rappresenta l'errore introdotto dalla semplificazione del modello rispetto alla vera funzione sottostante. Un modello con un alto bias è troppo semplice e può non catturare la complessità dei dati, portando a un errore sistematico. In altre parole, il bias è il differenziale tra il valore medio delle predizioni del modello e il valore reale.
- **Errore di Varianza:** La varianza rappresenta l'errore introdotto dalla sensibilità del modello alle fluttuazioni nel dataset di addestramento. Un modello con alta varianza si adatta troppo ai dati di addestramento e può avere performance scadenti su nuovi dati, portando a un'instabilità nelle predizioni. In altre parole, la varianza è la variabilità delle predizioni del modello per diversi set di addestramento.

Il trade-off bias-variance è essenziale per comprendere come il modello generalizza sui dati non visti. Idealmente, si cerca di bilanciare questi due errori per ottenere il miglior compromesso tra la capacità di adattamento e la generalizzazione.

#### 3.4.2 Trade-Off Bias-Variance in K-Nearest Neighbors

Nel contesto di KNN, il valore di  $K$  gioca un ruolo cruciale nel determinare il trade-off bias-varianza:

- **Basso Valore di  $K$  (Alta Complessità):** Quando  $K$  è molto basso (ad esempio,  $K = 1$ ), il modello di KNN è molto flessibile e si adatta strettamente ai dati di addestramento. Questo porta a una bassa bias e a una alta varianza. In altre parole, il modello ha una capacità di adattamento molto elevata ma rischia di sovradattarsi (overfitting) ai dati di addestramento, con prestazioni scadenti su nuovi dati non visti.
- **Valore Moderato di  $K$ :** Un valore moderato di  $K$  trova un equilibrio tra bias e varianza. In questa situazione, KNN considera un numero ragionevole di vicini per fare previsioni, riducendo l'errore di varianza senza aumentare eccessivamente l'errore di bias. Questo valore di  $K$  è spesso scelto attraverso tecniche di validazione incrociata per ottenere il miglior compromesso tra adattamento e generalizzazione.

- **Alto Valore di  $K$  (Bassa Complessità):** Quando  $K$  è molto alto, il modello di KNN diventa meno sensibile ai dati specifici e si adatta a una media dei vicini. Questo porta a un aumento del bias e a una diminuzione della varianza. Il modello diventa meno complesso e può generalizzare meglio su dati non visti, ma rischia di non catturare dettagli significativi nel dataset di addestramento, portando a un errore di bias più elevato e potenzialmente a un sottoadattamento (underfitting).

### 3.4.3 Selezione del Valore Ottimale di $K$

La scelta del valore ottimale di  $K$  è cruciale per gestire il trade-off bias-varianza. Ecco alcuni metodi per selezionare un valore appropriato:

- **Validazione Incrociata:** Utilizzare tecniche di validazione incrociata, come la  $k$ -fold cross-validation, per testare vari valori di  $K$  e selezionare quello che minimizza l'errore di generalizzazione sul set di validazione. In questa tecnica, il dataset viene diviso in  $k$ -folds (sottogruppi), e il modello viene addestrato e valutato  $k$  volte, ogni volta utilizzando un diverso fold come set di validazione e il resto come set di addestramento. La media degli errori di validazione per ciascun valore di  $K$  viene quindi utilizzata per selezionare il valore di  $K$  che minimizza l'errore.
- **Curva di Apprendimento:** Analizzare le curve di apprendimento per valori differenti di  $K$  può fornire indicazioni su come il bias e la varianza cambiano. Questo approccio può aiutare a visualizzare il trade-off e scegliere un valore di  $K$  che offre un buon compromesso tra overfitting e underfitting.
- **Euristiche:** In pratica, è comune utilizzare valori dispari per  $K$  per evitare ambiguità nella decisione di classificazione e iniziare con valori più piccoli e incrementare gradualmente per osservare come cambia la performance del modello.

In sintesi, il trade-off bias-variance in KNN è influenzato principalmente dal valore di  $K$ . Un valore troppo basso di  $K$  può portare a un'elevata varianza e a overfitting, mentre un valore troppo alto può portare a un alto bias e a underfitting. La selezione di un valore ottimale di  $K$  è essenziale per ottenere un buon equilibrio tra la capacità di adattamento e la generalizzazione del modello.

## 4 Ottimizzazioni

### 4.1 KD-Tree

### 4.2 Ball Tree

### 4.3 Hashing

### 4.4 Algoritmi Approximate Nearest Neighbors (ANN)

## Riferimenti bibliografici

- [1] Reinhard Diestel. *Graph Theory*. Springer Berlin Heidelberg, 2017.
- [2] J. A. Bondy and U. S. R. Murty. *Graph Theory*. Springer London, 2008.