Algoritme k-nearest neighbor (k-NN atau KNN) adalah sebuah metode untuk melakukan [klasifikasi](https://id.wikipedia.org/wiki/Pengenalan_pola" \o "Pengenalan pola) terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut.

Data pembelajaran diproyeksikan ke ruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan fitur dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi data pembelajaran. Sebuah titik pada ruang ini ditandai kelas c jika kelas c merupakan klasifikasi yang paling banyak ditemui pada k buah tetangga terdekat titk tersebut. Dekat atau jauhnya tetangga biasanya dihitung berdasarkan jarak Euclidean.

Pada fase pembelajaran, algoritme ini hanya melakukan penyimpanan vektor-vektor fitur dan klasifikasi dari data pembelajaran. Pada fase klasifikasi, fitur-fitur yang sama dihitung untuk data test (yang klasifikasinya tidak diketahui). Jarak dari vektor yang baru ini terhadap seluruh vektor data pembelajaran dihitung, dan sejumlah k buah yang paling dekat diambil. Titik yang baru klasifikasinya diprediksikan termasuk pada klasifikasi terbanyak dari titik-titik tersebut.

Nilai k yang terbaik untuk algoritme ini tergantung pada data; secara umumnya, nilai k yang tinggi akan mengurangi efek noise pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi lebih kabur. Nilai k yang bagus dapat dipilih dengan optimasi parameter, misalnya dengan menggunakan cross-validation. Kasus khusus di mana klasifikasi diprediksikan berdasarkan data pembelajaran yang paling dekat (dengan kata lain, k = 1) disebut algoritme nearest neighbor.

Ketepatan algoritme k-NN ini sangat dipengaruhi oleh ada atau tidaknya fitur-fitur yang tidak relevan, atau jika bobot fitur tersebut tidak setara dengan relevansinya terhadap klasifikasi. Riset terhadap algoritme ini sebagian besar membahas bagaimana memilih dan memberi bobot terhadap fitur, agar performa klasifikasi menjadi lebih baik.

Terdapat beberapa jenis algoritme pencarian tetangga terdekat, diantaranya:

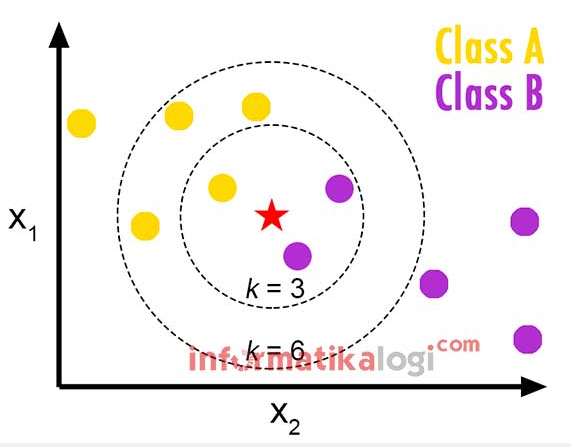
* Linear scan
* Pohon kd
* Pohon Balltree
* Pohon metrik
* Locally-sensitive hashing (LSH)

Ilustrasi cerita dari K-NN adalah sebagai berikut:

**Bertanya pada Tetangga** – Anda diundang ke sebuah pertemuan. Namun, Anda tidak tahu tema dari pertemuan tersebut, maupun kegiatan apa saja yang akan dilakukan di pertemuan tersebut. Anda benar-benar tidak tahu apakah pertemuan itu akan bermanfaat atau tidak untuk Anda. Yang Anda tahu, beberapa orang teman Anda juga diundang ke acara yang sama. Dalam kondisi seperti itu, apa yang Anda lakukan?

Cara yang biasanya dilakukan oleh banyak orang dalam menangani masalah seperti itu adalah dengan bertanya kepada teman-teman apakah mereka akan datang ke pertemuan tersebut atau tidak. Biasanya, orang-orang yang pertama ditanya adalah orang-orang yang dekat dengan Anda. Maka, Anda mencoba mengontak enam orang teman yang biasa jadi teman main Anda. Dari enam orang tersebut, empat orang menyatakan akan datang, tapi dua orang ternyata memutuskan tidak datang, entah mengapa alasannya. Keputusan apa yang Anda akan ambil?

Kasus di atas menggambarkan ide dari algoritma k-Nearest Neighbours (kNN). Anda ingin mengambil sebuah keputusan (kelas) antara datang atau tidak datang ke sebuah pertemuan. Untuk mendukung pengambilan keputusan tersebut, Anda melihat mayoritas dari keputusan teman-teman Anda (instance lainnya). Teman-teman tersebut Anda pilih berdasarkan kedekatannya dengan Anda. Ukuran kedekatan pertemanan ini bisa bermacam-macam: tetangga, satu hobi, satu kelas, atau hal-hal lainnya. Ukuran-ukuran tersebut bisa juga digunakan bersamaan, misalnya si A itu tetangga, satu hobi, dan satu kelas; sedangkan si B hanya satu kelas saja.



Gambar di atas menggambarkan ide dari algoritma **k-Nearest Neighbours (kNN)**. Anda ingin mengambil sebuah keputusan (kelas) antara datang atau tidak datang ke sebuah pertemuan. Untuk mendukung pengambilan keputusan tersebut, Anda melihat mayoritas dari keputusan teman atau tetangga Anda (instance lainnya). Teman atau tetangga tersebut Anda pilih berdasarkan kedekatannya dengan Anda. Ukuran kedekatan pertemanan ini bisa bermacam-macam: satu hobi, satu kelas, atau hal-hal lainnya. Ukuran-ukuran tersebut bisa juga digunakan bersamaan, misalnya si A itu tetangga, satu hobi, dan satu kelas; sedangkan si B hanya satu kelas saja. Dekat atau jauhnya tetangga biasanya dihitung berdasarkan Euclidean Distance, atau dapat juga menggunakan rumus jarak yang lain, seperti yang dijelaskan pada artikel [Vector Space Model dan Pengukuran Jarak](https://informatikalogi.com/vector-space-model-pengukuran-jarak/). Kedekatan dapat dianggap sebagai invers jarak, alias berbanding terbalik dengan jarak. Semakin kecil jarak antara dua instance, semakin besar “kedekatan” antara dua instance tersebut. Dengan demikian, k nearest neighbours dari sebuah instance x didefinisikan sebagai k instance yang memiliki jarak terkecil (kedekatan terbesar, nearest) dengan x.

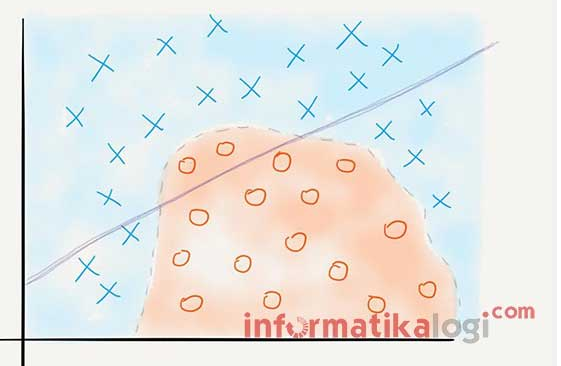
**2. Tahapan Langkah Algoritma K-NN**

1. Menentukan parameter k (jumlah tetangga paling dekat).
2. Menghitung kuadrat jarak eucliden objek terhadap data training yang diberikan.
3. Mengurutkan hasil no 2 secara ascending (berurutan dari nilai tinggi ke rendah)
4. Mengumpulkan kategori Y (Klasifikasi nearest neighbor berdasarkan nilai k)
5. Dengan menggunakan kategori nearest neighbor yang paling mayoritas maka dapat dipredisikan kategori objek.

**3. Kelebihan dan Kekurangan dari Algoritma K-NN**

**Kelebihan**

* **Sangat nonlinear**
  + kNN merupakan salah satu algoritma (model) pembelajaran mesin yang bersifat nonparametrik. Pembahasan mengenai **model parametrik** dan model **nonparametrik** bisa menjadi artikel sendiri, namun secara singkat, definisi model nonparametrik adalah model yang tidak mengasumsikan apa-apa mengenai distribusi instance di dalam dataset. Model nonparametrik biasanya lebih sulit diinterpretasikan, namun salah satu kelebihannya adalah garis keputusan kelas yang dihasilkan model tersebut bisa jadi sangat fleksibel dan nonlinear.



Pada ilustrasi di atas, kNN dapat melakukan klasifikasi dengan tepat karena garis keputusan kelasnya nonlinear. Bandingkan dengan model linear (**e.g. logistic regression**) yang tentunya akan menghasilkan banyak misklasifikasi jika garis keputusan kelas dalam dataset sebenarnya bersifat nonlinear.

* **Mudah dipahami dan diimplementasikan**
  + Dari paparan yang diberikan dan penjelasan cara menghitung jarak dalam artikel ini, cukup jelas bahwa algoritma kNN mudah dipahami dan juga mudah dimplementasikan. Untuk mengklasifikasi instance x menggunakan kNN, kita cukup mendefinisikan fungsi untuk menghitung jarak antar-instance, menghitung jarak x dengan semua instance lainnya berdasarkan fungsi tersebut, dan menentukan kelas x sebagai kelas yang paling banyak muncul dalam k instance terdekat.

**Kekurangan**

* **Perlu menunjukkan parameter K (jumlah tetangga terdekat)**
* **Tidak menangani nilai hilang (missing value) secara implisit**
  + Jika terdapat nilai hilang pada satu atau lebih variabel dari suatu instance, perhitungan jarak instance tersebut dengan instance lainnya menjadi tidak terdefinisi. Bagaimana coba, menghitung jarak dalam ruang 3-dimensi jika salah satu dimensi hilang? Karenanya, sebelum menerapkan kNN kerap dilakukan **imputasi** untuk mengisi nilai-nilai hilang yang ada pada dataset. Contoh teknik imputasi yang paling umum adalah mengisi nilai hilang pada suatu variabel dengan nilai rata-rata variabel tersebut (mean imputation).
* **Sensitif terhadap data pencilan (outlier)**
  + Seperti yang telah dijelaskan Ali pada artikel sebelumnya, kNN bisa jadi sangat fleksibel jika k kecil. Fleksibilitas ini mengakibatkan kNN cenderung sensitif terhadap data pencilan, khususnya pencilan yang terletak di “tengah-tengah” kelas yang berbeda. Lebih jelasnya, perhatikan ilustrasi di bawah. Pada gambar kiri, seluruh instance bisa diklasifikasikan dengan benar ke dalam kelas biru dan jingga. Tetapi, ketika ditambahkan instance biru di antara instance jingga, beberapa instance jingga menjadi salah terklasifikasi.Perlu dipilih k yang tepat untuk mengurangi dampak data pencilan dalam kNN.