Unsupervised Learning

by Thio Perdana

Outline

Pendahuluan

Unsupervised Learning atau Pembelajaran Tak Terarah adalah teknik pembelajaran mesin di mana pengguna tidak perlu mengawasi modelnya. Sebaliknya, pembelajaran tak terarah memungkinkan model bekerja sendiri untuk menemukan pola dan informasi yang sebelumnya tidak terdeteksi, terutama yang berkaitan dengan data yang tidak berlabel. Algoritma unsupervised learning memungkinkan pengguna untuk melakukan tugas pemrosesan yang lebih kompleks dibandingkan dengan supervised learning. Namun, pembelajaran tak terarah bisa lebih tidak terduga dibandingkan dengan metode pembelajaran

alami lainnya. Algoritma pembelajaran tak terarah termasuk pengelompokan, deteksi anomali, jaringan saraf, dan lain-lain.

Supervised Learning dan Unsupervised Learning adalah dua teknik pembelajaran mesin yang berbeda. Supervised Learning menggunakan data berlabel (labelled data) untuk melatih model, sedangkan Unsupervised Learning menggunakan data tanpa label (unlabeled data). Perbedaan utama antara Supervised Learning dan Unsupervised Learning adalah penggunaan data. Supervised Learning digunakan untuk tugas-tugas klasifikasi dan regresi, misalnya dalam kasus object recognition, predictive analysis dan sentiment analysis. Sedangkan Unsupervised Learning digunakan untuk kasus-kasus klastering, asosiasi dan dimensionality reduction.

Supervised Learning memerlukan data berlabel untuk melatih model. Data berlabel adalah data yang sudah diberi label atau kategori. Contohnya, jika kita ingin membuat model untuk mengenali gambar kucing, maka kita memerlukan data gambar kucing yang sudah diberi label “kucing”. Dalam kasus ini, model akan belajar dari data tersebut dan mencoba menemukan pola yang dapat digunakan untuk mengenali kucing pada gambar yang belum pernah dilihat sebelumnya.

Sedangkan Unsupervised Learning tidak memerlukan data berlabel. Model akan mencoba menemukan pola dan struktur dalam data yang tidak berlabel. Contohnya, jika kita memiliki data penjualan produk, kita dapat menggunakan Unsupervised Learning untuk mengelompokkan produk yang serupa ke dalam satu kelompok. Dalam kasus ini, model akan mencoba menemukan pola dan struktur dalam data penjualan produk dan mengelompokkan produk yang serupa ke dalam satu kelompok.

Beberapa contoh aplikasi Unsupervised Learning di kehidupan sehari-hari:

Pengelompokan: Unsupervised Learning dapat digunakan untuk mengelompokkan data yang tidak dikategorikan. Contohnya, dalam bidang pemasaran, Unsupervised Learning dapat digunakan untuk mengelompokkan pelanggan berdasarkan perilaku pembelian mereka. Dengan cara ini, perusahaan dapat menyesuaikan strategi pemasaran mereka untuk setiap kelompok pelanggan.

Deteksi anomali: Unsupervised Learning dapat digunakan untuk mendeteksi anomali dalam data. Contohnya, dalam bidang keamanan siber, Unsupervised Learning dapat digunakan untuk mendeteksi serangan siber yang tidak biasa atau tidak terduga.

Jaringan saraf: Unsupervised Learning dapat digunakan untuk mempelajari pola dalam data yang tidak terstruktur dan tidak berlabel. Contohnya, dalam bidang pengenalan wajah, Unsupervised Learning dapat digunakan untuk mempelajari pola dalam data wajah dan mengidentifikasi wajah yang serupa.

Jenis-Jenis Algoritma Unsupervised Learning

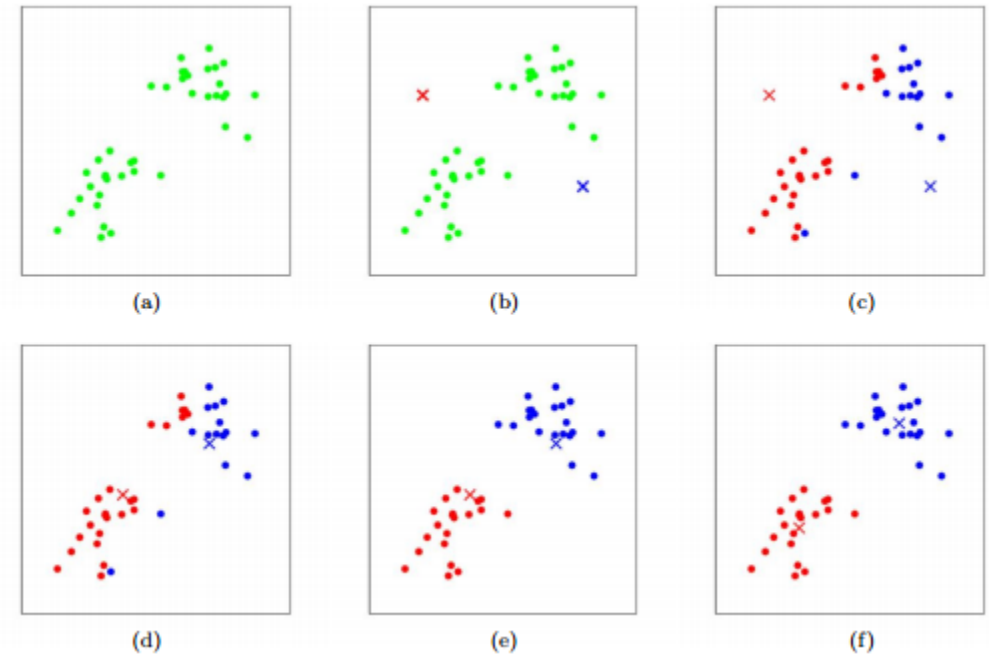
Mayoritas data yang beredar atau berhasil didapatkan merupakan data mentah yang belum memiliki label. Contohnya adalah data perjalan seseorang ke kantor. Kita mungkin akan mendapatkan banyak data misalnya 500 m, 700 m, dsb. Jika kita mendapatkan data 2 Km, mungkin kita akan melabelinya sebagai jarak tempuh jauh jika melihat data sebelumnya di 500 m. Akan tetapi, jika ternyata terdapat data dengan nilai 10 Km mungkin 2 Km tadi akan menjadi label jarak tempuh sedang. Dari hal tersebut terlihat betapa data yang akan kita dapatkan merupakan data mentah yang pelabelannya bisa sangat dtentukan oleh jumlah dataset yang kita miliki. Untuk mempelejari hal tersebut maka kita membutuhkan metode Unsupervised Learning

Clustering

Algoritma clustering pada unsupervised learning adalah suatu metode yang digunakan untuk mengelompokkan data menjadi beberapa kelompok atau kluster berdasarkan kemiripan fitur di antara elemen-elemen data. Tujuan utama dari algoritma clustering adalah untuk meminimalkan variasi intra-kluster dan memaksimalkan variasi antar-kluster.

K-Means

K-Means Clustering adalah salah satu algoritma unsupervised learning yang termasuk ke dalam analisis klaster (cluster analysis) non hirarki yang digunakan untuk mengelompokkan data berdasarkan variabel atau feature. Dalam K-Means Clustering, data dikelompokkan menjadi beberapa klaster berdasarkan jarak antara data terhadap titik centroid klaster yang didapatkan melalui proses berulang. Analisis perlu menentukan jumlah K sebagai input algoritma. Dalam ranah machine learning, algoritma K-Means Clustering termasuk ke dalam jenis unsupervised learning.



langkah-langkah k-means dilakukan. Sumber : https://stanford.edu/

Misalkan kita mempunyai data seperti pada gambar (a), jika kita menentukan bahwa kita akan menggunakan 2 sentroid maka akan di tempatkan dua buah sentroid (atau bisa disebut titik pusat massa) secara random. Nanti setiap data akan ditempatkan sesuai dengan nilai kedekatan dua buah atribut (x dan y) yang mereka miliki. Setelah itu kita akan mendapatkan dua zona klustering (gambar (c)). lalu kita akan mencari dua buah sentroid lagi, tapi kali ini tidak secara random tapi melihat dari setiap zona (gambar (d)). Setelah itu dilakukan lagi penempatan untuk mendapatkan zona klustering baru (gambar (e)). Ini terus dilakukan hingga didapatkan nilai sentroid yang stabil.

Contoh penggunaan K-Means Clustering antara lain:

Segmentasi pasar (market segmentation)

Segmentasi citra

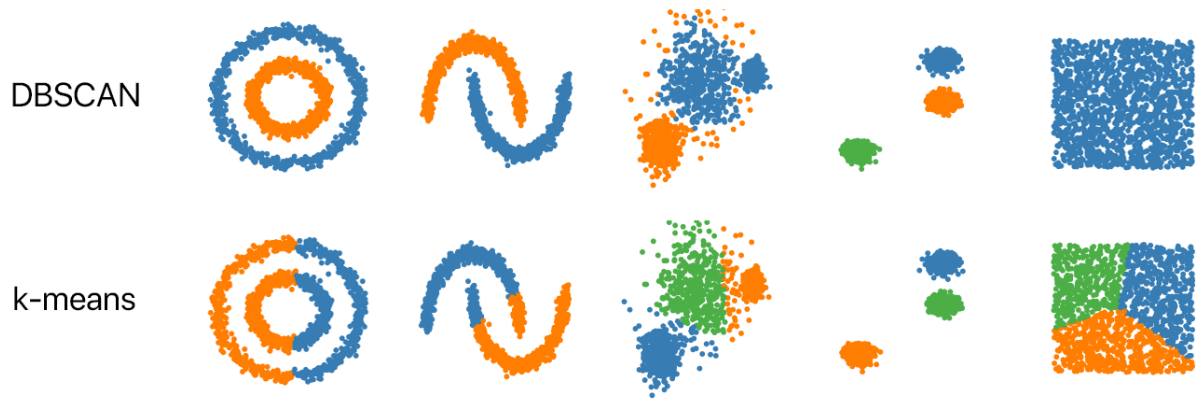
Kompresi gambar

Klasifikasi citra penginderaan jauh

DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) adalah salah satu algoritma unsupervised learning yang digunakan untuk mengelompokkan data berdasarkan kepadatan data. DBSCAN beroperasi berdasarkan konsep kerapatan data. Ini mencoba menemukan daerah yang memiliki tingkat kerapatan tinggi dan mengelompokkannya menjadi satu kluster. Kontras dengan algoritma clustering lain yang mungkin mengasumsikan bentuk kluster tertentu, DBSCAN dapat menangani kluster dengan bentuk dan ukuran yang berbeda. Fokus utama DBSCAN adalah mengidentifikasi kluster yang berdekatan secara spasial dalam data. Algoritma ini memandang data sebagai distribusi titik dalam ruang, dan kluster dibentuk oleh daerah dengan tingkat kerapatan yang

tinggi. DBSCAN dapat menangani data yang mungkin memiliki noise atau outlier. Noise ini tidak akan diikutsertakan dalam kluster dan dianggap sebagai kelompok terpisah.



Berikut adalah langkah-langkah umum dalam algoritma DBSCAN:

1. Tentukan nilai epsilon dan nilai minimum data point yang dibutuhkan untuk membentuk klaster.
2. Pilih titik acak sebagai titik awal dan cari semua titik yang berjarak kurang dari epsilon dari titik awal.
3. Jika jumlah titik yang ditemukan lebih besar dari nilai minimum data point, maka titik tersebut akan membentuk klaster baru.
4. Ulangi langkah 2 dan 3 untuk semua titik yang belum termasuk ke dalam klaster.

Contoh penggunaan DBSCAN antara lain:

Segmentasi pasar (market segmentation)

Segmentasi citra

Kompresi gambar

Klasifikasi citra penginderaan jauh

Keuntungan DBSCAN

Tidak memerlukan jumlah kluster sebagai input.

Dapat menangani kluster dengan bentuk yang kompleks dan ukuran yang berbeda.

Mengatasi noise atau outlier.

Kelemahan DBSCAN

Sensitif terhadap parameter `eps` dan `min_samples`.

Mungkin sulit untuk menangani data dengan kerapatan yang bervariasi secara signifikan.

DBSCAN adalah algoritma yang berguna untuk mengelompokkan data ke dalam kluster berdasarkan kerapatan dan dapat memberikan hasil yang baik untuk data yang tidak teratur atau memiliki tingkat kerapatan yang berbeda di berbagai bagian dataset.

Hierarchical Clustering

Hierarchical Clustering adalah metode dalam machine learning yang digunakan untuk mengelompokkan data ke dalam kelompok-kelompok yang disusun secara hierarki atau bertingkat. Bayangkan jika Anda ingin mengelompokkan keluarga besar Anda. Pertama, Anda dapat mengelompokkan mereka berdasarkan hubungan dekat (seperti saudara kandung atau sepupu), kemudian secara bertahap membentuk kelompok yang lebih besar berdasarkan kesamaan hubungan.

Hierarchical Clustering dibagi kedalam dua jenis utama, yaitu:

Agglomerative Clustering:

Dimulai dengan setiap titik dianggap sebagai kluster tunggal.

Iteratif menggabungkan kluster yang paling mirip satu sama lain.

Pada setiap iterasi, jarak antar-kluster dihitung dan dua kluster terdekat digabung.

Proses ini terus berlanjut hingga semua titik berada dalam satu kluster besar.

Divisive Clustering:

Dimulai dengan satu kluster besar yang berisi semua titik.

Iteratif memisahkan kluster menjadi dua kluster yang paling tidak mirip.

Pada setiap iterasi, kriteria pemisahan (misalnya, jarak maksimum) digunakan untuk memisahkan kluster.

Proses ini terus berlanjut hingga setiap titik berada dalam kluster terpisah.

Berikut adalah langkah-langkah umum dalam algoritma Hierarchical Clustering:

Metrik Kemiripan

Metrik kemiripan pada Hierarchical Clustering adalah cara untuk mengukur seberapa mirip atau berbedanya dua titik data atau kluster dalam konteks algoritma clustering. Metrik ini digunakan untuk menentukan jarak antar elemen data, yang selanjutnya memengaruhi proses penggabungan atau pemisahan kluster pada algoritma Hierarchical Clustering.

Terdapat berbagai metrik kemiripan yang dapat digunakan, dan pemilihan metrik ini dapat memengaruhi hasil clustering. Beberapa metrik kemiripan umum yang digunakan melibatkan perhitungan jarak antara dua titik data atau kluster. Beberapa di antaranya adalah:

Jarak Euclidean:

Definisi: Mengukur jarak langsung antara dua titik dalam ruang data.

Contoh: Jarak sejati antara dua titik di koordinat kartesian.

Jarak Manhattan (City Block):

Definisi: Mengukur jarak antara dua titik dengan menambahkan selisih absolut koordinat mereka.

Contoh: Jarak dalam kota antara dua lokasi di kota yang memiliki sistem jalan paralel.

Korelasi:

Definisi: Mengukur sejauh mana dua variabel berkorelasi satu sama lain.

Rumus: Bervariasi tergantung pada definisi korelasi yang digunakan (misalnya, Pearson, Spearman).

Contoh: Seberapa seragam hubungan antara dua variabel.

Jarak Minkowski:

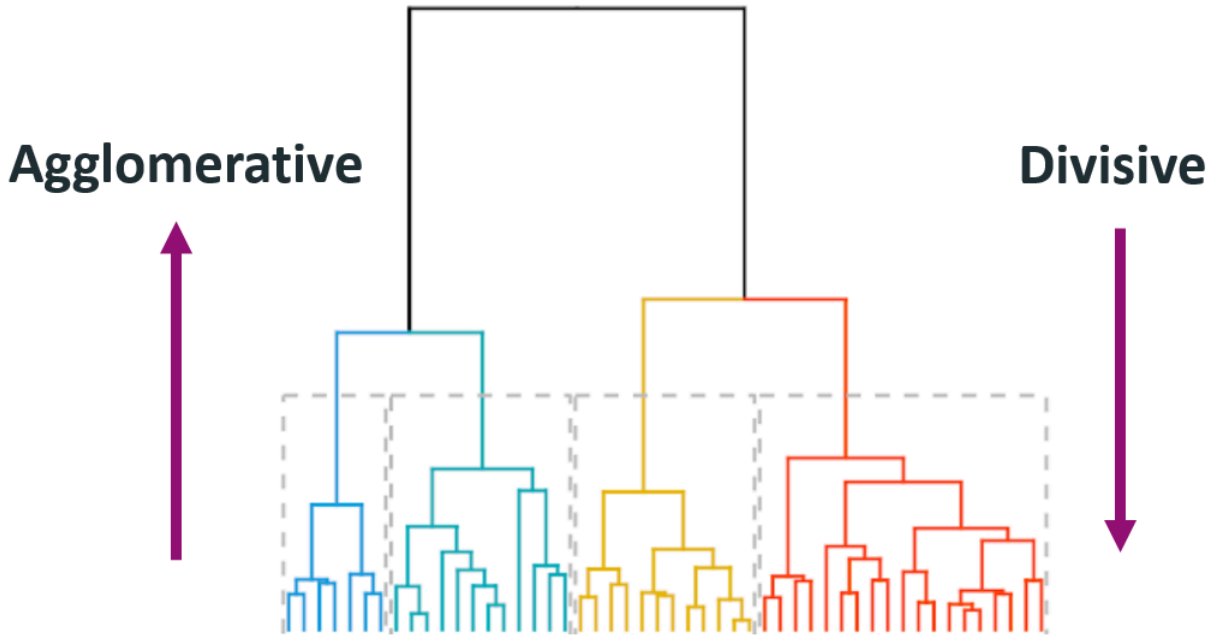
Definisi: Generalisasi dari jarak Euclidean dan Manhattan, diatur oleh parameter p.

Contoh: Ketika p=2, ini menjadi jarak Euclidean; ketika p=1, menjadi jarak Manhattan.

Pemilihan metrik kemiripan bergantung pada sifat data dan tujuan analisis. Beberapa metrik mungkin lebih sesuai untuk data tertentu atau mungkin menghasilkan struktur kluster yang lebih sesuai dengan kebutuhan analisis.

Dendrogram

Dendrogram adalah representasi grafis dari hasil dari proses Hierarchical Clustering. Dendrogram digunakan untuk menunjukkan hubungan hierarkis antara elemen-elemen data atau kluster dalam bentuk pohon. Struktur pohon ini memberikan gambaran visual tentang bagaimana kluster-kluster tersebut dibentuk dan bagaimana mereka terkait satu sama lain.



Berikut adalah komponen-komponen utama dalam dendrogram:

Garis-Garis Cabang (Branch Lines): Garis-garis ini menghubungkan elemen-elemen data atau kluster dan menunjukkan tahap-tahap pengelompokan atau pemisahan. Panjang garis mencerminkan seberapa jauh dua elemen atau kluster tersebut terpisah.

Node (Node): Setiap simpul pada dendrogram mewakili satu atau beberapa elemen data atau kluster. Pada awalnya, setiap elemen data adalah simpul sendiri, dan selama proses clustering, simpul-simpul ini bergabung untuk membentuk kluster yang lebih besar.

Titik Potong (Cutting Point): Pada dendrogram, titik potong menandai tingkat di mana kita memilih untuk memotong atau mengakhiri pembentukan kluster. Memotong dendrogram pada suatu tingkat tertentu menghasilkan kluster-kluster yang sesuai dengan tingkat tersebut.

Dendrogram memberikan informasi visual yang berguna tentang struktur hierarkis data. Pengamatan yang berdekatan dalam dendrogram cenderung memiliki kemiripan yang lebih tinggi, sedangkan pengamatan yang lebih jauh dapat memiliki kemiripan yang lebih rendah. Dengan memotong dendrogram pada tingkat tertentu, kita dapat membentuk kluster-kluster yang sesuai dengan kebutuhan analisis atau interpretasi data.

Keputusan Pemotongan (Cutting Dendrogram)

Keputusan pemotongan (cutting dendrogram) pada Hierarchical Clustering adalah langkah di mana kita memilih untuk memotong atau mengakhiri proses pembentukan kluster pada dendrogram pada tingkat tertentu. Dendrogram adalah representasi grafis dari hasil clustering hierarkis, dan dengan memotong dendrogram pada suatu tingkat, kita dapat membentuk kluster-kluster yang sesuai dengan tingkat tersebut.

Pemotongan dendrogram merupakan langkah kritis dalam analisis Hierarchical Clustering, dan tingkat pemotongan yang dipilih akan memengaruhi seberapa banyak dan seberapa besar kluster yang dihasilkan. Pemotongan ini dapat dilakukan dengan menggunakan beberapa kriteria, antara lain:

Jarak atau Ketinggian (Height): Pemotongan dapat dilakukan berdasarkan tinggi (jarak) tempat dua kluster bergabung. Tingkat pemotongan yang lebih tinggi menghasilkan kluster yang lebih besar, sementara tingkat pemotongan yang lebih rendah menghasilkan kluster yang lebih kecil.

Jumlah Kluster (Number of Clusters): Sebagai alternatif, pemotongan dapat dilakukan dengan menentukan jumlah kluster yang diinginkan. Misalnya, kita dapat memilih untuk membentuk lima kluster, dan algoritma akan memotong dendrogram sedemikian rupa sehingga terbentuk lima kluster.

Skor Keseragaman (Cophenetic Correlation Coefficient): Metrik ini dapat digunakan untuk menilai seberapa baik pemotongan dendrogram merefleksikan kesamaan dalam data. Semakin tinggi skor keseragaman, semakin baik pemotongan tersebut.

Keputusan pemotongan ini dapat sangat memengaruhi interpretasi hasil clustering, dan pemilihan metode pemotongan harus didasarkan pada pemahaman tugas analisis dan sifat data. Beberapa analisis eksploratif dan visualisasi dendrogram dapat membantu peneliti untuk menentukan tingkat pemotongan yang paling relevan untuk tujuan analisis mereka.

Hierarchical clustering cocok digunakan ketika Anda ingin memahami struktur hierarki di dalam data dan mendapatkan gambaran yang lebih mendalam tentang kemiripan antar elemen data. Pemahaman ini dapat membantu dalam pengelompokan yang lebih intuitif dan pemahaman hierarki dalam data.

Association

Association merupakan salah satu jenis dari Unsupervised Learning yang berfokus pada penemuan hubungan atau asosiasi antara item atau atribut dalam data. Dalam Association Learning, algoritma mencoba untuk menemukan pola-pola tersembunyi atau korelasi antara item atau atribut tanpa adanya label atau output yang sudah diketahui sebelumnya. Salah satu algoritma yang umum digunakan dalam Association Learning adalah Apriori. Misalnya, dalam sebuah supermarket, Association Learning dapat

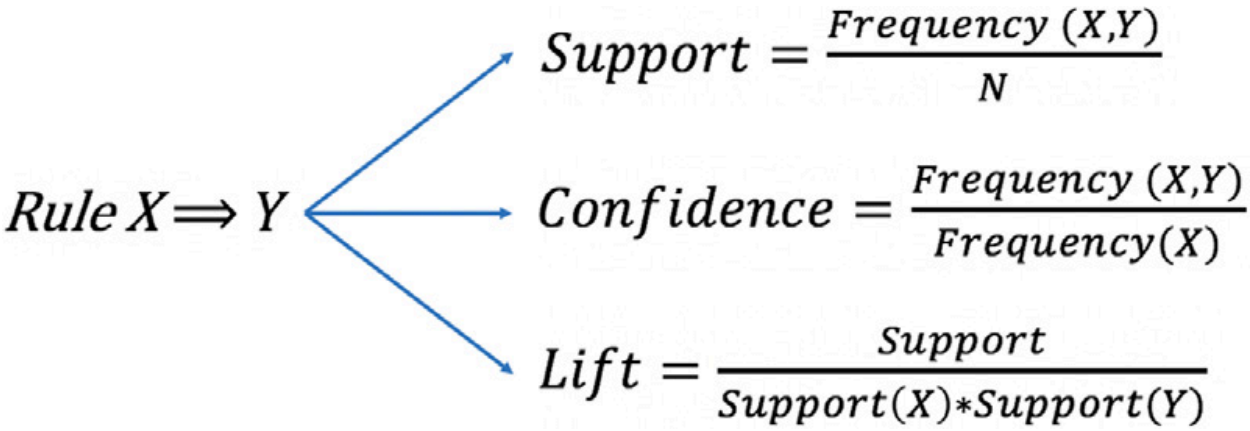
digunakan untuk menemukan pola pembelian bersamaan, seperti "Jika seseorang membeli roti, kemungkinan besar juga akan membeli mentega."

Beberapa istilah penting dalam algoritma apriori, yaitu

itemset Itemset adalah kumpulan satu atau lebih item atau atribut. Misalnya, {roti, susu} adalah itemset yang terdiri dari roti dan susu.

Support Support mengukur seberapa sering sebuah itemset muncul dalam dataset. Jika banyak transaksi yang mengandung itemset tersebut, support-nya tinggi.

Confidence Confidence mengukur seberapa sering aturan asosiasi ditemukan benar. Misalnya, "Jika seseorang membeli roti, seberapa sering mereka juga membeli susu?" Confidence mengukur seberapa kuat hubungan ini.



Algoritma Apriori

Association Learning, khususnya menggunakan algoritma Apriori, adalah teknik dalam Unsupervised Learning yang bertujuan untuk menemukan hubungan atau asosiasi antara item atau atribut dalam

dataset tanpa adanya label atau output yang telah diketahui sebelumnya. Algoritma ini berasal dari gagasan bahwa jika suatu itemset memiliki support tinggi, maka setiap subset dari itemset tersebut juga harus memiliki support tinggi.

Langkah-langkah umum menggunakan algoritma Apriori, yaitu:

Inisialisasi Itemsets: Tentukan dataset belanjaan dan tentukan nilai minimum support yang diinginkan. Minimum support adalah batas persentase atau jumlah transaksi di mana sebuah itemset dianggap "frequent".

Hitung Frekuensi Itemset 1-item: Hitung frekuensi kemunculan masing-masing item (produk) pada dataset. Itemset yang memenuhi minimum support akan menjadi kandidat frequent itemset.

Gabungkan Itemset untuk Itemset K+1: Dari frequent itemset 1-item yang telah diidentifikasi, gabungkan itemset yang serupa untuk membentuk kandidat frequent itemset 2-item. Kemudian, lakukan hal yang sama untuk mendapatkan kandidat frequent itemset 3-item, dan seterusnya.

Hitung Frekuensi Kandidat Itemset: Hitung frekuensi kemunculan kandidat frequent itemset pada dataset.

Filter Frequent Itemset: Hapus kandidat yang tidak memenuhi minimum support dari list kandidat. Hasilnya adalah frequent itemset yang sesuai dengan minimum support.

Ulangi Langkah 3-5 hingga Tidak Ada Kandidat Lagi: Lanjutkan langkah-langkah 3 hingga 5 untuk setiap kategori itemset hingga tidak ada kandidat frequent itemset baru yang dapat dihasilkan.

Selesaikan Proses: Jika sudah tidak mungkin menghasilkan kandidat frequent itemset baru, selesaikan proses dan hasilkan frequent itemset terakhir yang memenuhi minimum support.

Hasil Akhir: Hasilkan frequent itemset yang memenuhi minimum support dan digunakan untuk mendukung analisis asosiasi dan rekomendasi.

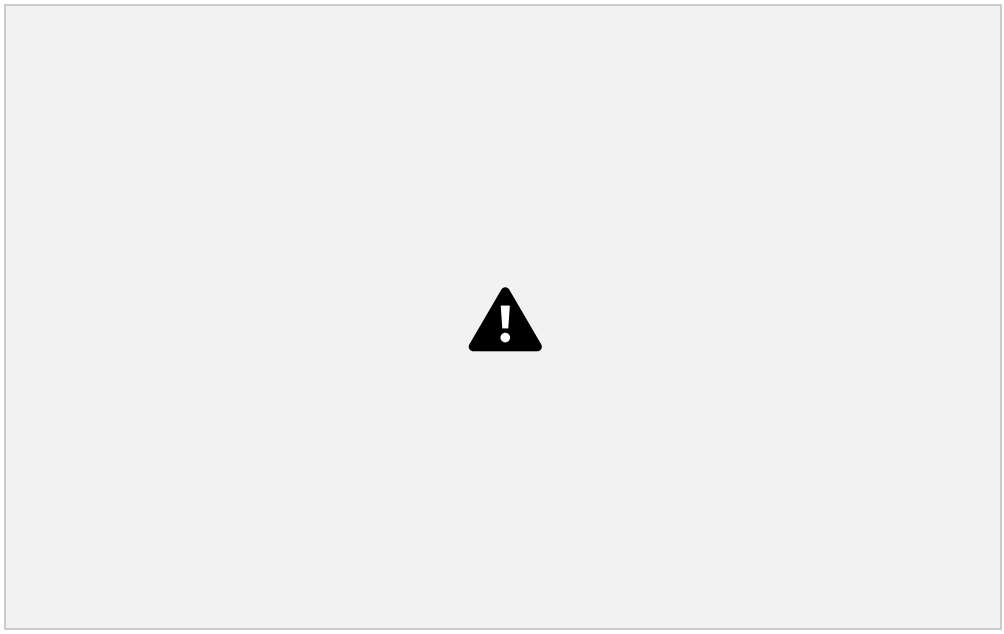
Rule Association: Dari frequent itemset yang telah diidentifikasi, hasilkan aturan asosiasi dengan menetapkan tingkat kepercayaan (confidence) sebagai threshold. Aturan ini memberikan informasi tentang seberapa sering itemset A diikuti oleh itemset B.

Langkah-langkah ini membantu mengidentifikasi pola pembelian yang sering muncul dalam dataset belanjaan, yang dapat digunakan untuk memberikan rekomendasi produk atau menyusun strategi pemasaran yang lebih efektif.

Algoritma FP-growth

FP-growth (Frequent Pattern growth) adalah algoritma untuk menemukan itemset yang sering muncul dalam dataset transaksi. Ini digunakan dalam analisis asosiasi, terutama untuk menemukan pola pembelian yang sering terjadi.

Langkah-langkah dan konsep utama dari algoritma FP-growth:



Scan Pertama (First Scan):

Hitung frekuensi kemunculan setiap item tunggal dalam dataset transaksi.

Hapus item yang tidak memenuhi minimum support (batas frekuensi minimum).

Bentuk Pohon Pertumbuhan Frequent Pattern (FP-tree):

Susun dataset transaksi ke dalam struktur data FP-tree.

Itemset yang sering muncul diurutkan berdasarkan frekuensi mereka dalam dataset.

Itemset yang sering muncul di setiap transaksi digunakan untuk membangun FP-tree.

Pohon Pertumbuhan (Tree Growth):

Setiap transaksi digunakan untuk membangun jalur dalam FP-tree.

Itemset yang sering muncul di transaksi ditempatkan pada jalur yang sesuai dalam FP-tree.

Jika jalur sudah ada, tambahkan frekuensi itemset yang sesuai.

Pencarian Pattern Frequent (Frequent Pattern Mining):

Lakukan pencarian secara rekursif untuk menemukan pattern frequent pada setiap cabang FP-tree.

Kombinasikan hasil dari setiap cabang untuk mendapatkan frequent pattern yang lebih besar.

Generate Aturan Asosiasi:

Gunakan frequent pattern yang telah ditemukan untuk menghasilkan aturan asosiasi.

Aturan ini memberikan informasi tentang seberapa sering itemset A diikuti oleh itemset B.

Keunggulan FP-growth dibandingkan dengan metode lain seperti Apriori adalah bahwa algoritma ini hanya memerlukan dua kali pemindaian (scan) dataset. Selain itu, FP-growth menggunakan struktur data FP-tree yang memungkinkan pencarian pola yang lebih efisien.

FP-growth cocok untuk dataset dengan jumlah transaksi yang besar, dan khususnya efektif ketika sebagian besar itemset pada dataset tidak frequent. Dengan kata lain, FP-growth dapat menangani dataset yang memiliki ruang pencarian besar dengan lebih efisien.

Dimensionality Reduction

Algoritma reduksi dimensi adalah teknik yang digunakan dalam pembelajaran tanpa pengawasan untuk mengurangi jumlah fitur atau variabel dalam dataset tanpa kehilangan informasi penting. Pemahaman tentang algoritma reduksi dimensi dapat membantu memproses dan menganalisis data yang kompleks dengan lebih efisien. Berikut adalah beberapa algoritma reduksi dimensi yang umum digunakan:

Principal Component Analysis (PCA)

Tujuan: Mengidentifikasi arah di mana data memiliki variasi maksimal.

Proses: Menghitung vektor eigen dan nilai eigen dari matriks kovariansi data.

Keuntungan: Mengurangi dimensi data dengan mempertahankan sebagian besar variasi.

Keterbatasan: Komponen utama tidak selalu mudah diinterpretasikan secara intuitif.

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Tujuan: Memetakan data ke dalam dimensi rendah untuk mempertahankan struktur jarak lokal.

Proses: Mengukur probabilitas distribusi kemiripan antar-pasangan data dalam dimensi tinggi dan rendah.

Keuntungan: Efektif untuk memvisualisasikan struktur kluster atau kelompok dalam data.

Keterbatasan: Rentan terhadap variasi dalam inisialisasi dan mungkin tidak mempertahankan jarak global.

Linear Discriminant Analysis (LDA)

Tujuan: Mencari proyeksi yang memaksimalkan jarak antar-klasifikasi dan meminimalkan dispersi dalam setiap kelas.

Proses: Menghitung matriks kovariansi dalam dan antar-klasifikasi.

Keuntungan: Membantu meningkatkan pemisahan antar-klasifikasi.

Keterbatasan: Memerlukan informasi label kelas dalam data.

Algoritma-algoritma ini membantu mengatasi masalah high-dimensional data dengan menyaring fitur yang kurang relevan atau redundan, sehingga meningkatkan efisiensi pemrosesan dan memungkinkan pemahaman yang lebih baik tentang struktur data. Pemilihan algoritma tergantung pada karakteristik data dan tujuan analisisnya.

Implementasi

Klustering - KMeans

Persiapan Data

Data yang akan kita gunakan adalah data pokemon yang diambil menggunakan metode webscraping.

pokemonUnduh

Mencari Zona Klustering

Menggunakan data yang telah kita dapatkan kita akan mencoba untuk melakukan klusterisasi pada data tersebut. Sebagai contoh kali ini kita akan menggunakan metode K-Means dengan nilai K adalah 3.

Pertama kita membaca data yang dihasilkan pada bagian sebelumnya, kita akan menggunakan bantuan pandas untuk memudahkan.

```
import pandas as pd

data = pd.read_csv("data_pokemon.csv")
# Preview the first 5 lines of the loaded data

print(data.head())
```

Untuk melakukan klustering kita membutuhkan dua atribut yang akan menjadi basis perhitungan nilai mereka nanti. Pada kesempatan kali ini kita akan menggunakan nilai Attack dan Defense, karena menurut penulis tumbuh kembang pokemon akan berbanding lurus dengan Attack dan Defense mereka.

Sebelumnya kita akan memastikan data yang kita miliki merupakan numeric dan kita akan menambahkan kolom transformasi untuk kedua atribut tersebut. Transformasi yang akan saya gunakan adalah logaritmik. Jangan lupa untuk menambahkan library numpy pada library sebelumnya

```
   data["Attack"] = pd.to_numeric(data["Attack"])
    data["Defense"] = pd.to_numeric(data["Defense"])

    #menambahkan dua kolom tranformasi
    data["Alog"] = np.log(data["Attack"])
    data["Dlog"] = np.log(data["Defense"])


    print(data.head())
```

Setelah itu kita akan mengambil data atribut yang kita pakai dan mengubahnya ke dalam bentuk array.

```
   log_data = data.iloc[:, 10:12]
    print(log_data.head())
    #       Alog      Dlog
    #0  3.891820  3.891820
    #1  4.127134  4.143135
    #2  4.406719  4.418841
    #3  4.605170  4.812184
    #4  3.951244  3.761200

    # mengubahnya menjadi array
    log_array = np.array(log_data)
    print (log_array)
    #[[3.8918203  3.8918203 ]
    # [4.12713439 4.14313473]
    # [4.40671925 4.41884061]

    # [4.60517019 4.81218436]...]
```

Sekarang mari kita mulai untuk melakukan klustering, kita akan menggunakan library sklearn untuk itu dan library matplotlib untuk melakukan visualisasi. Tambahkan kedua library tersebut di awal.

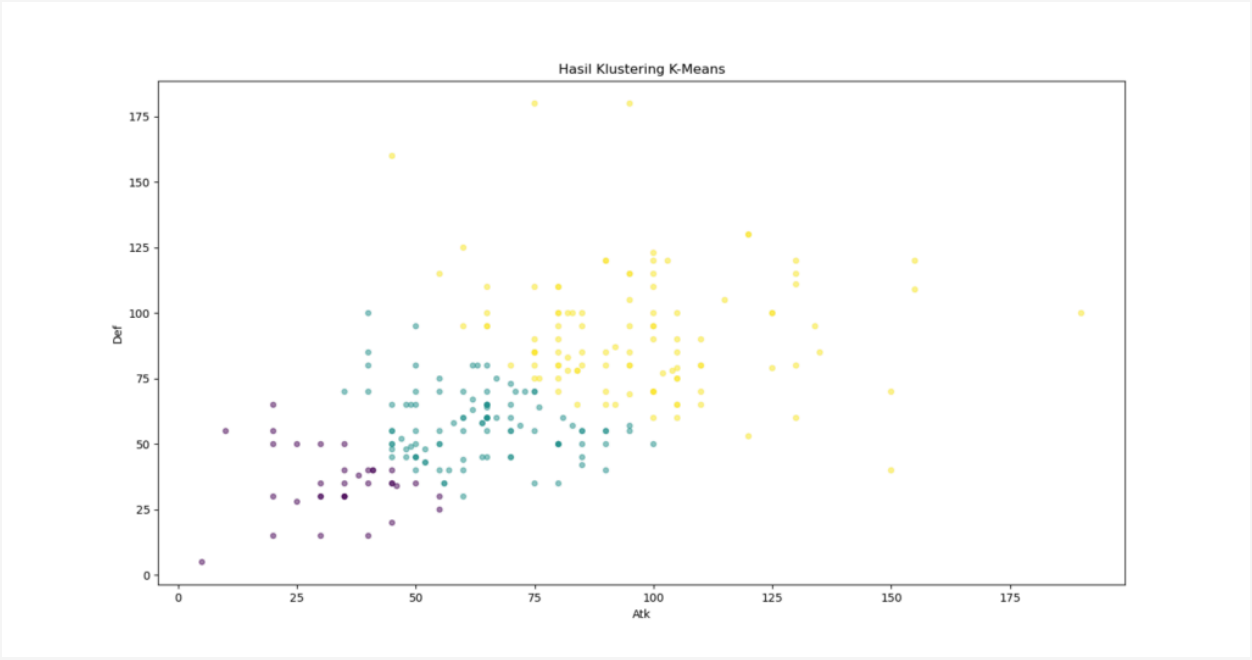
```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

kmeans = KMeans(n_clusters=3, random_state=200)
kmeans.fit(log_array)
data['kluster'] = kmeans.labels_

print(data.head())
```

variabel kmeans digunakan untuk menginisiasi metode kita dengan n menunjukan jumlah sentroid dan random state sebagai dasar dari pengacakan saat melakukan pengolahan data. kmeans.fit digunakan untuk meemulai melakukan pengolahan dan hasilnya disimpan kembali ke dalam dataframe awal.

Selanjutnya kita akan melakukan visualisasi agar dapat melihat data kita lebih baik.

```
plt.scatter(data.Attack, data.Defense, s = 20, c = data.kluster, marker = "o", alpha
= 0.5)
plt.title("Hasil Klustering K-Means")
```

```
plt.xlabel("Atk")
plt.ylabel("Def")

plt.show()
```



hasil visualisasi

Mencari 'K'

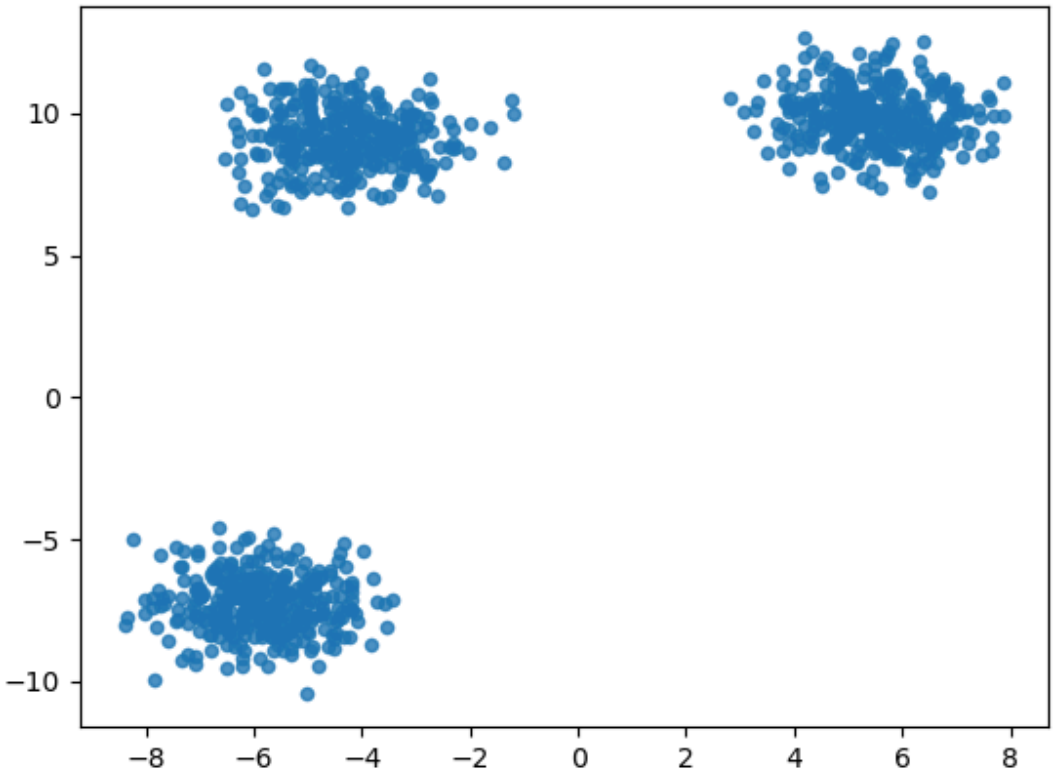
Seperti pada namanya, penentuan nilai K merupakan salah satu hal penting untuk dilakukan. Pada contoh sebelumnya kita telah terlebih dahulu menentukan nilai k yang akan kita pakai adalah 3. Pertanyaannya apakah ini merupakan nilai terbaik? untuk dapat menjawabnya kita akan menggunakan dua metode penentuan nilai k terbaik, yaitu *Elbow Method* dan *Silhouette Method.* Kita akan menggunakan data yang kita generate agar lebih mudah membayangkan.

Sekarang mari kita persiapkan datanya.

```
from sklearn.datasets import make_blobs
```

```
X, y = make_blobs(n_samples=1000, centers=3, n_features=2, shuffle=True,
random_state=31)
plt.scatter(X[:, 0], X[:, 1], s = 20, marker = "o", alpha = 0.8)

plt.show()
```

Maka data akan terlihat seperti,



hasil visualisasi

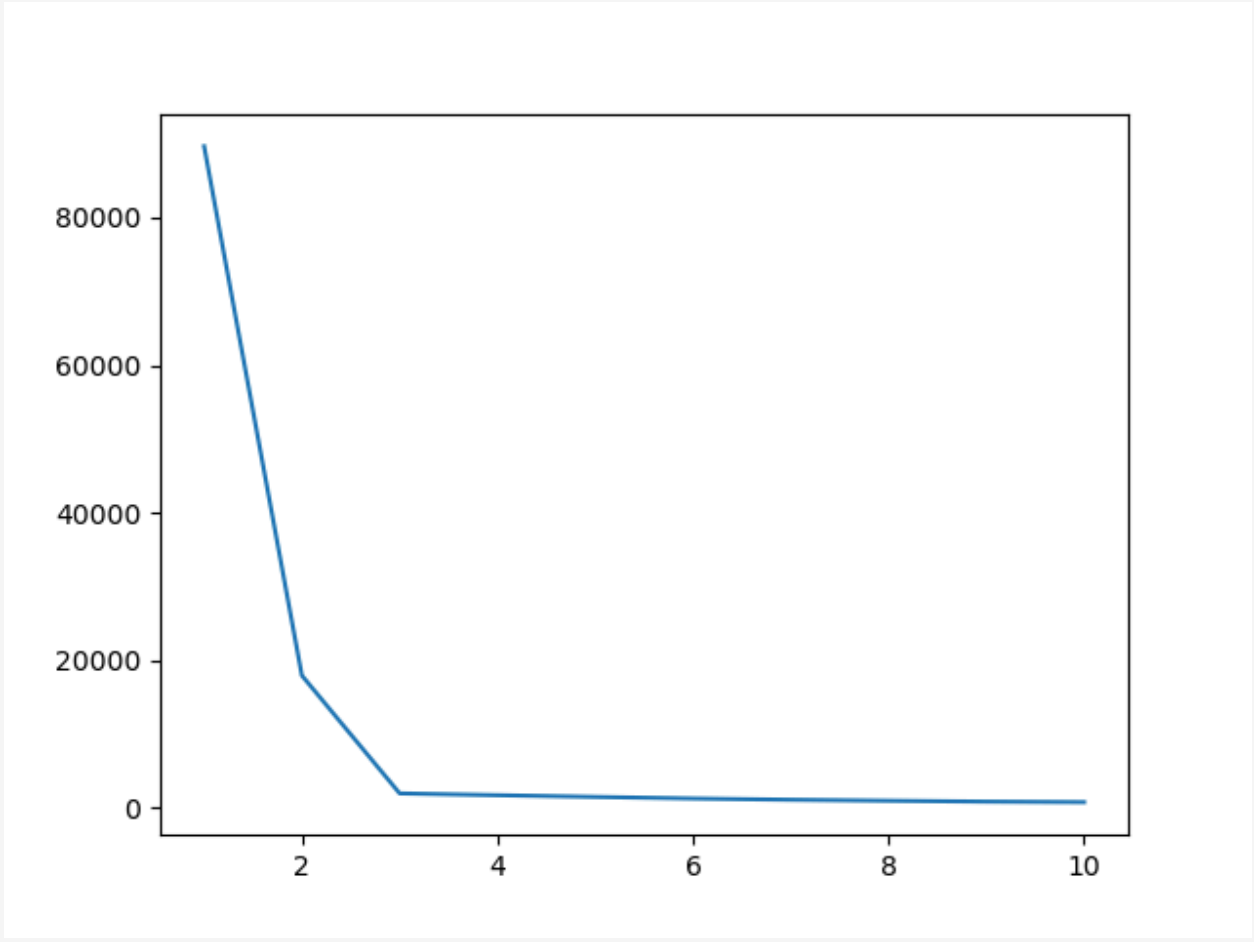
Elbow Method

Elbow Method menggunakan apa yang dinamakan kuadrat jarak, dimana jarak yang dimaksud disni adalah jarak antar titik dengan pusat klusternya dimana nanti ini akan dijumlahkan dan dinamakan WSS (Within-Cluster-Sum of Squared). Nilai K terbaik didapatkan dimana nilai WSS sudah mulai stabil, atau saat membentuk 'siku'.

```
sse = []
k_list = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters = k).fit(X)
    centroids = kmeans.cluster_centers_
    prediksi = kmeans.predict(X)
    nilai_sse = 0

    for i in range(len(X)):
        titik_pusat = centroids[prediksi[i]]
        nilai_sse += (X[i, 0] - titik_pusat[0]) ** 2 + (X[i, 1] - titik_pusat[1]) **
2

    sse.append(nilai_sse)

    k_list.append(k)
```

Lalu kita lakukan plot,

```
   plt.plot(k_list,sse)

    plt.show()
```



hasil plot

Dari Plot tersebut kita mengetahui nilai terbaik adalah nilai k = 3 , dimana dari data sebelumnya ini terbukti. Akan tetapi, tidak setiap waktu kita dapat dengan jelas melihat dimana keberadaan titik belok dari plot. Pada saat itulah kita akan menggunakan metode lain.

Silhouette Method

Silhouette Method menggunakan nilai koefisien yang dihitung dari seberapa dekat relasi antara objek dalam sebuah cluster, dan seberapa jauh sebuah cluster terpisah dengan cluster lain. persamaan yang digunakan adalah,

Koefisien *Silhouette* = *(x-y)/ max(x,y)*

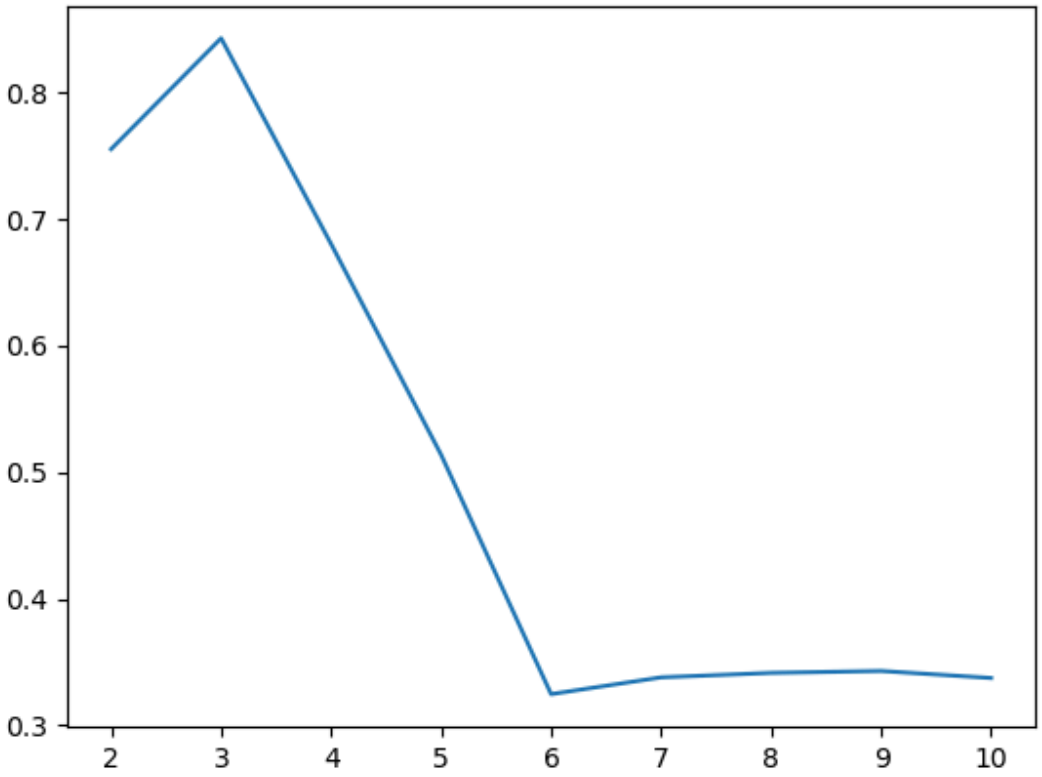
Dimana x adalah jarak dengan kluster lain dan y adalah jarak relasi antar objek pada kluster yang sama.

Nilai K optimum didapatkan dari nilai puncak plot K terhadap Koefisien *Silhouette*

Kita akan membutuhkan modul tambahan dari sklearn untuk menggunakan metode ini.

```
from sklearn.metrics import silhouette_score

data = []
k_list = []

for k in range(2, 11):
    kmeans = KMeans(n_clusters = k).fit(X)
    labels = kmeans.labels_
    data.append(silhouette_score(X, labels, metric = 'euclidean'))
    k_list.append(k)

plt.plot(k_list,data)

plt.show()
```

Maka akan didapatkan hasil,



hasil visualisasi

Dari hasil tersebut kita dapat melihat bahwa nilai puncak berada pada nilai K = 3 dan ini sesuai dengan data yang kita miliki.

Association - Apriori

Pada kesempatan kali ini kita akan menggunakan dataset pembelian sederhana

```
# Data belanjaan

data_belanjaan = [
    ['Roti', 'Susu', 'Telur'],
    ['Roti', 'Keju'],
    ['Roti', 'Mentega', 'Telur'],
    ['Susu', 'Keju', 'Telur'],
    ['Roti', 'Susu'],
    ['Roti', 'Keju', 'Telur'],
    ['Roti', 'Susu', 'Mentega'],
    ['Keju', 'Mentega'],
    ['Roti', 'Susu', 'Keju'],
    ['Telur', 'Mentega'],
    ['Susu', 'Keju', 'Mentega'],
    ['Roti', 'Telur'],
    ['Roti', 'Susu', 'Keju', 'Mentega', 'Telur'],
    ['Susu', 'Keju'],
    ['Roti', 'Susu', 'Mentega', 'Telur'],
    ['Roti', 'Keju', 'Mentega'],
    ['Susu', 'Telur'],
    ['Roti', 'Susu', 'Keju'],
    ['Keju', 'Mentega', 'Telur'],
    ['Roti', 'Susu', 'Keju', 'Telur'],
```

```
    ['Roti', 'Susu', 'Keju', 'Mentega'],
    ['Roti', 'Keju', 'Mentega', 'Telur'],
    ['Roti', 'Keju', 'Telur'],
    ['Susu', 'Mentega', 'Telur'],
    ['Roti', 'Susu', 'Mentega', 'Keju'],
    ['Roti', 'Mentega'],
    ['Keju', 'Telur'],
    ['Roti', 'Susu', 'Keju', 'Mentega', 'Telur'],
    ['Roti', 'Keju', 'Telur'],
    ['Roti', 'Susu', 'Mentega']
]
```

Untuk kebutuhan analisis ini kita akan menggunakan library mlextend. Jika menggunakan environment lokal, silahkan install terlebih dahulu dengan menggunakan perintah

```
pip install mlextend
```

Selanjutnya kita panggil semua library yang dibutuhkan

```
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
import pandas as pd
```

Langkah selanjutnya, kita akan mengubah data kita kedalam bentuk transaksi, atau vektor item

```
te = TransactionEncoder()

te_ary = te.fit(data_belanjaan).transform(data_belanjaan)

df = pd.DataFrame(te_ary, columns=te.columns_)
```

Yang mana jika kita tampilkan hasilnya akan menjadi

	Keju	Mentega	Roti	Susu	Telur
0	False	False	True	True	True
1	True	False	True	False	False
2	False	True	True	False	True
3	True	False	False	True	True
4	False	False	True	True	False
5	True	False	True	False	True
6	False	True	True	True	False
7	True	True	False	False	False
8	True	False	True	True	False
9	False	True	False	False	True
10	True	True	False	True	False
11	False	False	True	False	True

12	True	True	True	True	True

Lalu kita mencari frequent itemsets dengan algoritma Aprior, dan tampilkan

```
frequent_itemsets = apriori(df, min_support=0.2, use_colnames=True)
```

```
# Menampilkan frequent itemsets
print("Frequent Itemsets:")
display(frequent_itemsets)
```

	support	itemsets
0	0.633333	(Keju)
1	0.533333	(Mentega)
2	0.700000	(Roti)
3	0.566667	(Susu)
4	0.566667	(Telur)
5	0.300000	(Mentega, Keju)
6	0.433333	(Roti, Keju)
7	0.333333	(Keju, Susu)

8	0.333333	(Telur, Keju)
9	0.366667	(Mentega, Roti)
10	0.300000	(Mentega, Susu)
11	0.266667	(Mentega, Telur)

Hasil di atas merupakan semua item/kombinasi item yang punya nilai support di atas 0.2. Setelah ini kita hanya perlu mencari aturan asosiasinya. kita tetapkan batasnya di 0.7

```
# Mencari asosiasi aturan dengan confidence >= 0.7
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)

# Menampilkan asosiasi aturan
print("\nAssociation Rules:")
display(rules)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(Susu)	(Roti)	0.566667	0.7	0.400000	0.705882	1.008403	3.333333e-03	1.02	1.923077e-02
1	(Keju, Susu)	(Roti)	0.333333	0.7	0.233333	0.700000	1.000000	2.775558e-17	1.00	1.784287e-16

2	(Telur, Keju)	(Roti)	0.333333	0.7	0.233333	0.700000	1.000000	2.775558e-17	1.00	1.784287e-16
3	(Mentega, Susu)	(Roti)	0.300000	0.7	0.233333	0.777778	1.111111	2.333333e-02	1.35	1.428571e-01

Support menunjukkan popularitas rata-rata produk atau item dalam dataset

Confidence mengacu pada kemungkinan seorang pelanggan membeli susu dan roti secara bersamaan.

(untuk kasus pertama)

Lift adalah peningkatan rasio penjualan susu dan roti dibandingkan jika kita menjual secara terpisah.