ntainer( margin:

.dren:

styl

child:

Google Developer Groups

On Campus • Telkom University Bandung

# Machine Learning 101

(Unsupervised Learning)







ardava-barus



@rdavaa



Muhammad Karov Ardava Barus Machine Learning Mentor @GDGoC Tel-U,

Data Science Student | Telkom University

### **Presentation Link**



ristek.link/ML-SLIDES-5



# **Today's Topic**

#### **Core Concepts**

- Definition
- Main Algorithms
- Applications
- Challenges

#### Clustering

- Definition
- K-Means
- Agglomerative Clustering
- DBSCAN

#### **Dimensionality Reduction**

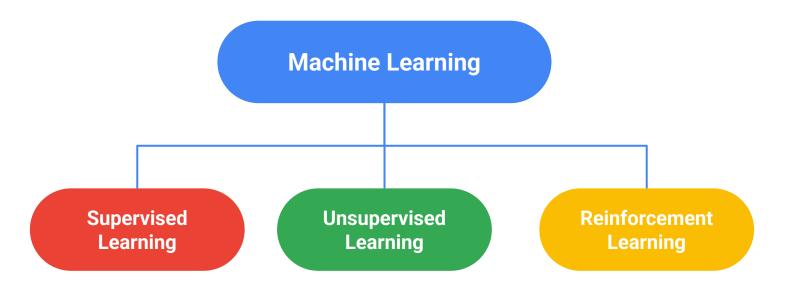
- Definition
- PCA
- t-SNE

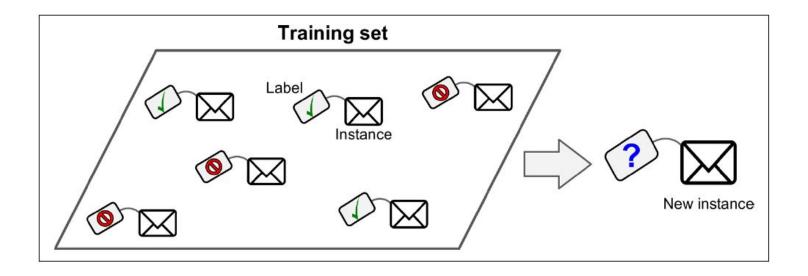
ookup.KeyValu .constant(['e tf.constant([ lookup.Static



# Let's recap Supervised Learning!

ookup.KeyValu .constant(['e tf.constant([ lookup.Static **Recap:** Supervised Learning





**Recap:** Supervised Learning

### Regression

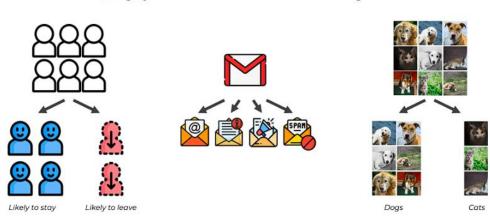
Dalam Machine Learning, regresi adalah metode untuk memprediksi nilai kontinu dari variabel dependen berdasarkan hubungan dengan variabel independen.



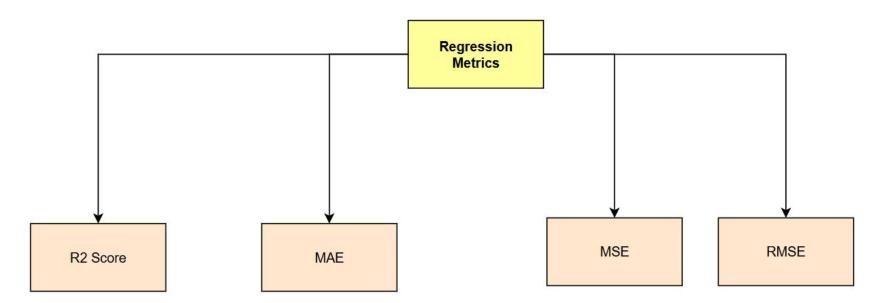
**Recap:** Supervised Learning

#### Classification

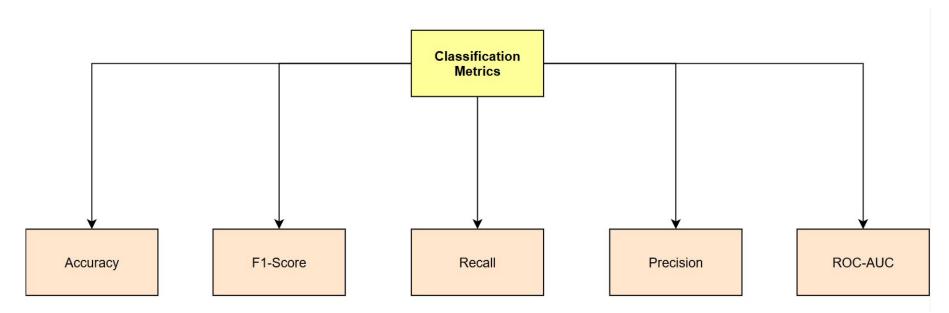
Dalam Machine Learning, klasifikasi adalah metode untuk memprediksi kategori atau kelas dari variabel dependen berdasarkan hubungan dengan variabel independen. Classification: a Machine Learning technique to identify the <a href="mailto:category">category</a> of new observations based on training data.



### **Evaluation Metrics - Regression**



#### **Evaluation Metrics - Classification**

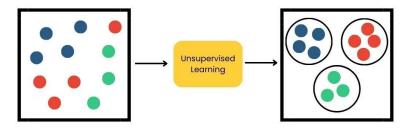




# **Core Concepts**

lookup.KeyValu f.constant(['e =tf.constant([ .lookup.Statio buckets=5)

# What is Unsupervised Learning?



- Unsupervised Learning adalah teknik machine learning yang digunakan untuk menemukan pola atau struktur tersembunyi dalam data tanpa adanya label atau panduan sebelumnya.
  - Unsupervised: Data tidak memiliki atribut target
  - Supervised: Data memiliki atribut target

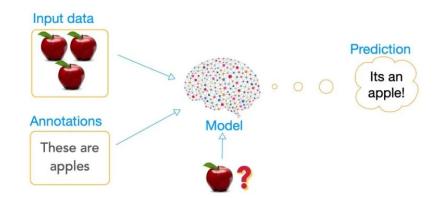
#### **Core Concepts:** Definition

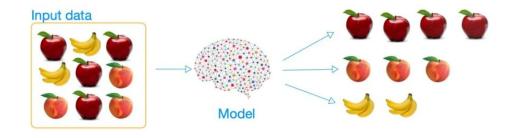
**Supervised Learning** (e.g. Regression, Classification)



**Unsupervised Learning** (e.g. Clustering)

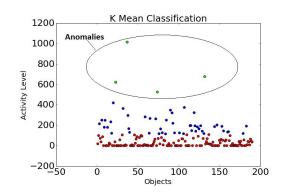






#### **Core Concepts:** Definition

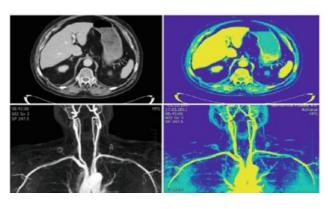
### **Real Life Applications**



**Anomaly Detection** 



**Customer Segmentation** 

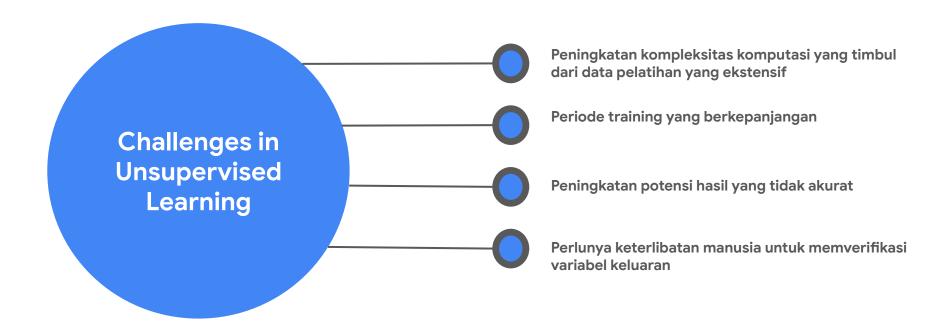


**Image Segmentation** 

and more...!



**Core Concepts:** Challenges





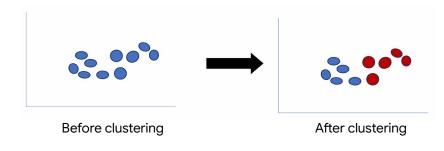
Clustering

# **Definition**

Lookup.KeyValu f.constant(['e =tf.constant([ .lookup.Static

### What is Clustering?

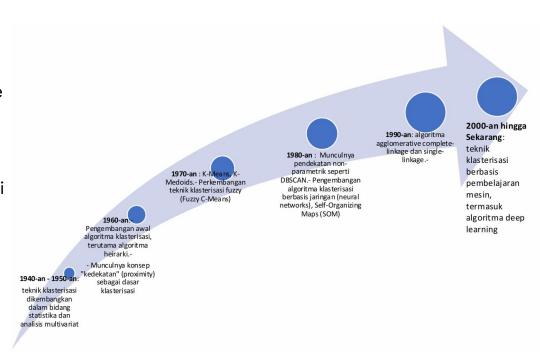
- Clustering adalah teknik pengelompokan data yang memiliki karakteristik serupa ke dalam kelompok-kelompok atau klaster.
- Tujuan utama dari klasterisasi adalah untuk menemukan pola tersembunyi dalam data tanpa menggunakan label kelas sebelumnya.



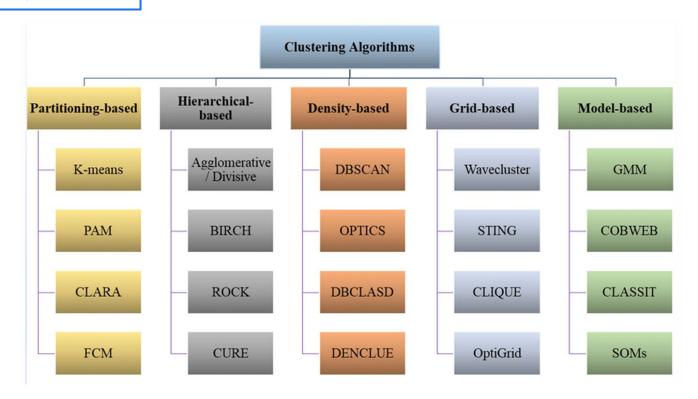
Illustration

### Sejarah Clustering

- Clustering telah berkembang dari metode statistika sederhana menjadi serangkaian algoritma canggih dan dapat menangani berbagai jenis data.
- Penggunaannya yang luas dalam berbagai bidang ilmu pengetahuan dan industri, membuatnya menjadi salah satu teknik analisis data yang paling penting dan relevan hingga saat ini.



**Clustering:** Definition



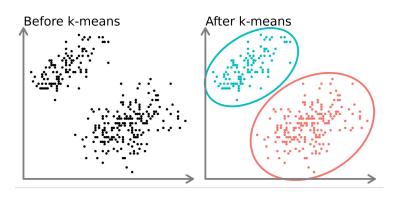


Clustering

# **K-Means**

ookup.KeyValu constant(['e tf.constant(| lookup.Statio

### **K-Means**



 K-Means adalah algoritma clustering yang bertujuan membagi n titik data ke dalam k kelompok (klaster) berdasarkan jarak ke pusat klaster (centroid).

Setiap klaster diwakili oleh rata-rata (mean) dari titik-titik di dalamnya, sehingga dinamakan "K-Means".

#### Kelebihan

- Sederhana dan cepat untuk dataset besar.
- Efektif jika klaster berbentuk bulat dan terpisah dengan baik.

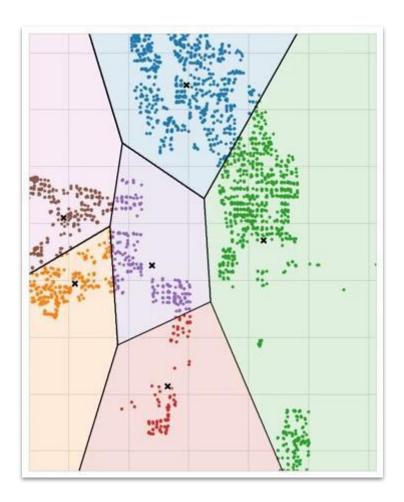
#### Kekurangan

- Harus menentukan k secara manual.
- Rentan terhadap *outliers*.
- Tidak cocok untuk klaster dengan bentuk tidak teratur

#### Clustering: K-Means

#### Algorithm 1 k-means clustering

```
1: Initialise Cluster Centers
 2: for each iteration l do
      Compute r_{nk}:
      for each data point x_n do
         Assign each data point to a cluster:
         for each cluster k do
 6:
           if \mathbf{k} == \mathrm{argmin} \|\mathbf{x}_n - \boldsymbol{\mu}_k^{l-1}\| then
              r_{nk} = 1
            else
              r_{nk} = 0
            end if
10:
11:
         end for
12:
      end for
13:
14:
      for each cluster k do
         Update cluster centers as the mean of each cluster:
15:
      end for
18: end for
```



#### **Model Initialization**

```
from sklearn.preprocessing import StandardScaler
    from sklearn.cluster import KMeans
    # feature scaling
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    # initialize and fit K-means model
    kmeans = KMeans(
       n_clusters=3, # number of clusters
        init='k-means++', # initialization method
        n_init='auto', # number of initializations
        max_iter=300, # maximum number of iterations
        random_state=42 # random seed
    kmeans.fit(X_scaled)
```

#### **Elbow Method in K-Means**

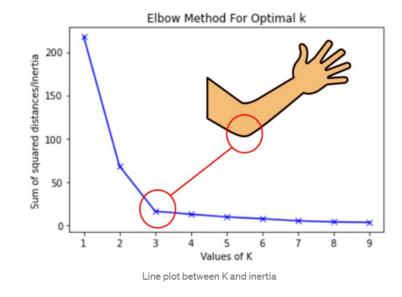
Elbow Method adalah teknik untuk menentukan jumlah klaster optimal (k) dalam algoritma K-Means

#### Kelebihan

- Sederhana dan intuitif untuk diinterpretasikan.
- Memberikan panduan visual untuk memilih *k*.
- Cocok untuk dataset dengan klaster yang jelas.

#### Kekurangan

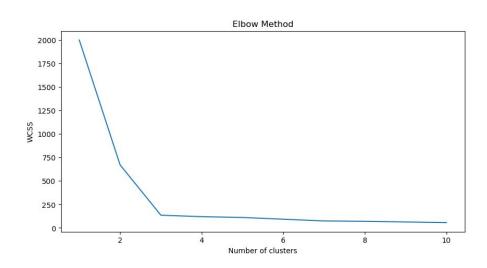
 Tidak selalu ada "siku" yang jelas, terutama pada data kompleks atau klaster tidak terpisah baik



#### **Elbow Method in K-Means**

```
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(10, 5))
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



From the figure above, the optimal number of clusters based on the elbow method visualization is 3.

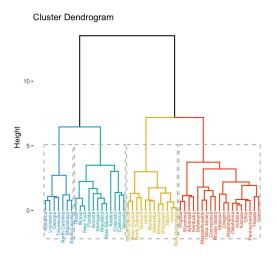


Clustering

# **Agglomerative Clustering**

.ookup.KeyValue f.constant(['en =tf.constant([@ .lookup.Static\ buckets=5) Clustering: Agglomerative Clustering

# **Agglomerative Clustering**



 Agglomerative Clustering adalah cara mengelompokkan data dengan pendekatan dari bawah ke atas.

Setiap titik data mulai sebagai kelompok sendiri, lalu kelompok yang mirip digabung bertahap hingga jadi satu atau sesuai jumlah yang diinginkan.

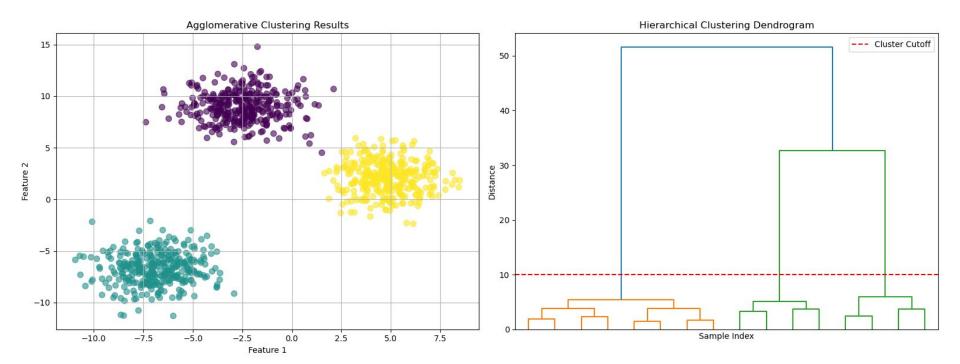
#### Kelebihan

- Cocok untuk data kecil hingga sedang dan klaster dengan bentuk tidak teratur.
- Tidak perlu tentukan k di awal (fleksibel dengan dendrogram).

#### Kekurangan

- Kompleksitas algoritma yang tinggi
- Keputusan penggabungan tidak bisa dibatalkan (greedy algorithm).

## Clustering: Agglomerative Clustering



#### **Model Initialization**

```
from sklearn.cluster import AgglomerativeClustering
    from sklearn.preprocessing import StandardScaler
    # feature scaling (consistent with previous example)
    scaler = StandardScaler()
   X scaled = scaler.fit transform(X)
    # initialize and fit Agglomerative Clustering
    agg cluster = AgglomerativeClustering(
        n clusters=3,
                             # number of clusters
       linkage='ward',
                             # linkage criteria
        compute full tree=True # for dendrogram analysis
15 cluster labels = agg cluster.fit predict(X scaled)
```

#### **Dendogram Visualization**

```
1 from scipy.cluster.hierarchy import dendrogram, linkage
 2 import matplotlib.pyplot as plt
 4 # create linkage matrix
 5 linkage matrix = linkage(
        X scaled,
       method='ward', # Consistent with clustering linkage
       metric='euclidean'
11 # configure dendrogram plot
12 plt.figure(figsize=(12, 6))
13 dendrogram(
        linkage matrix.
       truncate_mode='level', # Simplify visualization
                             # Truncation depth
       p=3.
       show leaf counts=True, # Show cluster sizes
       orientation='top',
                             # Orientation of tree
       labels=agg cluster.labels
22 plt.axhline(y=10, color='r', linestyle='--', label='Cluster Cutoff')
23 plt.title('Hierarchical Clustering Dendrogram')
24 plt.xlabel('Sample Index')
25 plt.ylabel('Distance')
26 plt.legend()
27 plt.show()
```



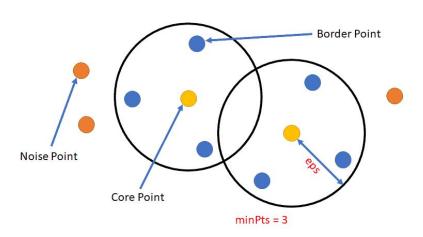
Clustering

# **DBSCAN**

lookup.KeyVal f.constant([' =tf.constant( .lookup.Stati

#### **Clustering: DBSCAN**

## **DBSCAN**



 DBSCAN adalah algoritma pengelompokan (clustering) berbasis kepadatan yang dirancang untuk mengidentifikasi kelompok data dalam ruang fitur berdasarkan distribusi kepadatan titik-titik.

Setiap titik data mulai sebagai kelompok sendiri, lalu kelompok yang mirip digabung bertahap hingga jadi satu atau sesuai jumlah yang diinginkan.

#### Kelebihan

- Tidak perlu tentukan jumlah klaster (k).
- Mampu menangani klaster dengan bentuk tidak beraturan (misalnya spiral atau lingkaran).
- Secara eksplisit mendeteksi outlier, cocok untuk data noisy.

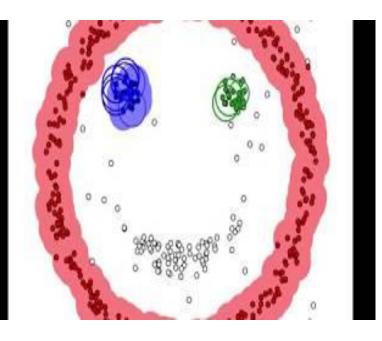
#### Kekurangan

- Kompleksitas algoritma yang tinggi
- Performa menurun pada data berdimensi sangat tinggi karena "curse of dimensionality" (jarak jadi kurang bermakna).

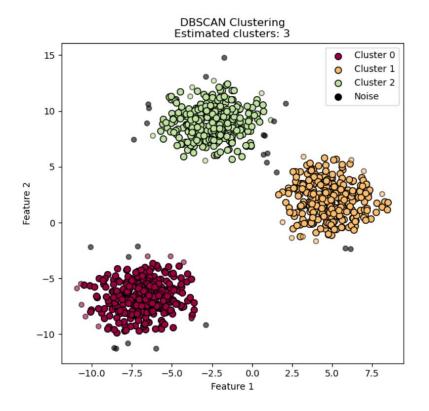
#### **Clustering: DBSCAN**

#### ALGORITHM 1: Pseudocode of Original Sequential DBSCAN Algorithm

```
Input: DB: Database
   Input: \varepsilon: Radius
   Input: minPts: Density threshold
   Input: dist: Distance function
   Data: label: Point labels, initially undefined
1 foreach point p in database DB do
                                                                              // Iterate over every point
       if label(p) \neq undefined then continue
                                                                              // Skip processed points
       Neighbors N \leftarrow \text{RangeQuery}(DB, dist, p, \varepsilon)
                                                                              // Find initial neighbors
3
       if |N| < minPts then
                                                                              // Non-core points are noise
            label(p) \leftarrow Noise
5
            continue
       c \leftarrow \text{next cluster label}
                                                                              // Start a new cluster
       label(p) \leftarrow c
       Seed set S \leftarrow N \setminus \{p\}
                                                                              // Expand neighborhood
       for each q in S do
10
            if label(q) = Noise then label(q) \leftarrow c
11
            if label(q) \neq undefined then continue
12
            Neighbors N \leftarrow \text{RangeQuery}(DB, dist, q, \varepsilon)
13
            label(q) \leftarrow c
14
            if |N| < minPts then continue
                                                                              // Core-point check
15
            S \leftarrow S \cup N
```



#### **Model Initialization**





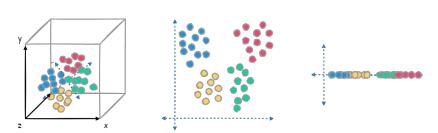


**Dimensionality Reduction** 

# **Definition**

Lookup.KeyValu constant(['etf.constant(| lookup.Statio Dimensionality Reduction:
Definition

# What is Dimensionality Reduction?



**Dimensionality Reduction** 

- Dimensionality Reduction adalah proses
   mengurangi jumlah dimensi (fitur atau variabel)
   dalam dataset sambil mempertahankan sebanyak
   mungkin informasi penting.
- Tujuannya adalah menyederhanakan data agar lebih mudah dianalisis, divisualisasikan, atau diproses oleh model machine learning tanpa kehilangan makna utama.
- Mengapa Penting?
  - Efisiensi
  - Noise Reduction
  - Curse of Dimensionality: Data berdimensi tinggi seringkali sulit diproses, membutuhkan lebih banyak komputasi, dan rentan terhadap overfitting.



## A great example to describe Dimensionality Reduction

## Dimensionality reduction





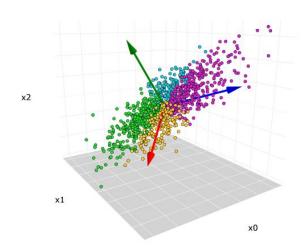
**Dimensionality Reduction** 

# Principal Component Analysis (PCA)

lookup.KeyValue constant(['en tf.constant([@ lookup.Static\ buckets=5)

### **Dimensionality Reduction: PCA**

## **PCA**





 PCA adalah teknik untuk reduksi dimensi linier yang digunakan untuk mengubah data berdimensi tinggi menjadi representasi dengan dimensi lebih rendah sambil mempertahankan sebanyak mungkin variasi (informasi) dalam data

#### Kelebihan

- Tidak perlu tentukan jumlah klaster (k).
- Mampu menangani klaster dengan bentuk tidak beraturan (misalnya spiral atau lingkaran).
- Secara eksplisit mendeteksi outlier, cocok untuk data noisy.

#### Kekurangan

- Linearitas: PCA adalah teknik linier dan mungkin tidak dapat menangkap hubungan nonlinier yang kompleks dalam data.
- Sensitif terhadap skala data: harus dinormalisasi terlebih dahulu.
- Hilangnya Informasi: Meskipun PCA mempertahankan sebagian besar varians, ada beberapa informasi yang hilang ketika mengurangi dimensi.

## Initialization

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import numpy as np
# data standardization (critical for PCA)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# initialize and fit PCA
pca = PCA(n_components=2) # keep top 2 components
principal_components = pca.fit_transform(X_scaled)
```



**Dimensionality Reduction** 

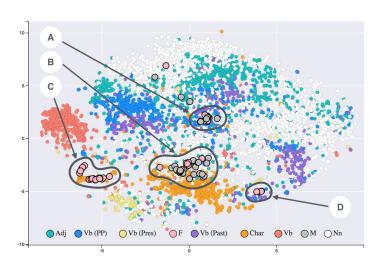
# t-distributed Stochastic Neighbor Embedding (t-SNE)

ookup.KeyValue constant(['entrologiestricker] tf.constant([@lookup.Staticker]

\_buckets=5)

**Dimensionality Reduction: t-SNE** 

## t-SNE



 t-SNE adalah teknik reduksi dimensi non-linier yang digunakan terutama untuk visualisasi data berdimensi tinggi (seperti ribuan fitur) ke ruang dimensi rendah, biasanya 2D atau 3D.

### Kelebihan

- Unggul dalam mempertahankan struktur lokal dan menangani data non-linier.
- Menghasilkan visualisasi yang intuitif untuk data kompleks (misalnya gambar, teks, embedding).
- Mampu menangkap klaster dengan bentuk tidak beraturan.

#### Kekurangan

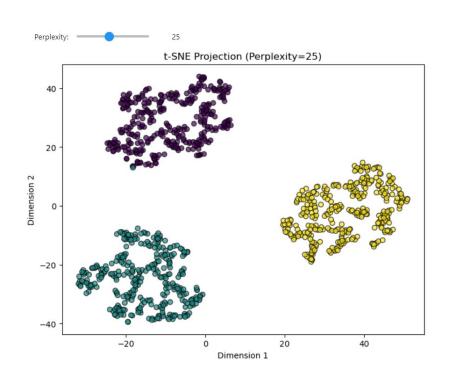
- Sangat lambat untuk dataset besar
- Tidak cocok untuk feature extraction atau prediksi (hanya visualisasi).

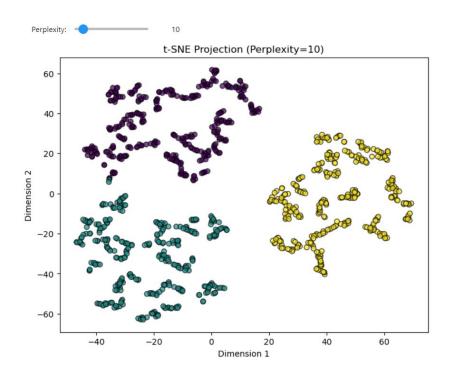
### Initialize t-SNE

```
from sklearn.manifold import TSNE
   # data standardization (recommended for t-SNE)
    scaler = StandardScaler()
   X_scaled = scaler.fit_transform(X)
    # initialize and fit t-SNE
   tsne = TSNE(
                                # Output dimensionality
       n components=2,
       perplexity=30,
                                # Default: 30, try values between 5-50
       early exaggeration=12.0, # Controls cluster spacing
       learning_rate='auto',  # Auto-adjust based on data size
       n_iter=1000,
                                # Minimum iterations
       random_state=SEED,
       n_jobs=-1
16 )
   tsne_embeddings = tsne.fit_transform(X_scaled)
```

## **Dimensionality Reduction: t-SNE**

## Initialize t-SNE









## Hands On

**Dataset** 



ristek.link/marketing-data22

Lookup.KeyValu f.constant(['e =tf.constant([ .lookup.Statio



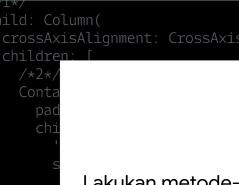
## **Demo Source Code**



ristek.link/qdqoc-demo-unsupervised

lookup.KeyValı f.constant(['e =tf.constant([ .lookup.Static

\_buckets=5)



## Tugas

Lakukan metode-metode Clustering atau Dimensionality Reduction menggunakan metode unsupervised learning yang sudah dibahas, seperti K-Means, DBSCAN, Agglomerative, PCA, atau t-SNE.

Analisis hasilnya untuk mendapatkan insight yang bermakna, seperti pola kelompok, hubungan antar data, atau karakteristik utama dataset.

Minimal pengerjaan adalah 3 dataset!

Source: ristek.link/ML-penugasan-5



Google Developer Groups



## Let's Connect!

• Instagram: @rdavaa\_

• LinkedIn: <u>www.linkedin.com/in/ardava-barus</u>