Laporan Tugas:
Analisis PCA dan MDS pada Dataset Iris

Identitas Mahasiswa

Nama: Thalita Zahra Sutejo

NIM: 18222023

Kelas: 01 - *Data Analytics*

1. Pendahuluan

Dalam era digital saat ini, pertumbuhan data yang sangat cepat telah menghasilkan dataset dengan jumlah fitur atau atribut yang sangat besar. Data seperti ini dikenal sebagai data berdimensi tinggi (*high-dimensional data*). Contoh umum dari data berdimensi tinggi meliputi citra digital, genomik, teks, dan data sensor. Meskipun banyaknya fitur dapat memberikan informasi yang kaya, hal ini juga menimbulkan sejumlah tantangan dalam proses analisis, visualisasi, dan pemodelan.

Salah satu tantangan utama dalam menganalisis data berdimensi tinggi adalah fenomena yang dikenal sebagai *curse of dimensionality*, yaitu ketika meningkatnya jumlah dimensi menyebabkan data menjadi sangat tersebar dan jarang, sehingga sulit untuk menemukan pola yang bermakna. Selain itu, model pembelajaran mesin cenderung mengalami overfitting ketika dilatih pada data berdimensi tinggi karena jumlah parameter yang besar dapat mempelajari noise alih-alih pola sejati. Proses komputasi juga menjadi lebih mahal dan tidak efisien.

Oleh karena itu, reduksi dimensi menjadi salah satu teknik penting dalam analisis data. Tujuan utamanya adalah menyederhanakan representasi data dengan mengurangi jumlah fitur tanpa kehilangan informasi yang relevan. Reduksi dimensi memungkinkan visualisasi yang lebih mudah, mempercepat pemrosesan data, dan meningkatkan kinerja model. Dua metode yang banyak digunakan dalam konteks ini adalah ***Principal Component Analysis*** **(PCA)** dan ***Multidimensional Scaling*** **(MDS)**. Kedua teknik ini digunakan untuk

memproyeksikan data ke ruang berdimensi lebih rendah, sehingga hubungan dan pola antar data dapat lebih mudah dianalisis dan divisualisasikan.

2. Metodologi

Bab ini menjelaskan secara rinci **langkah-langkah teknis** yang dilakukan dalam proses analisis data menggunakan teknik *dimensionality reduction*, yaitu ***Principal Component Analysis*** **(PCA)** dan ***Multidimensional Scaling*** **(MDS)**. Metodologi ini mencakup tiga aspek utama, yaitu **implementasi algoritma**, **alat dan bahasa pemrograman yang digunakan**, serta **tahapan *preprocessing* data** sebelum reduksi dimensi dilakukan. Proses ini bertujuan untuk memastikan bahwa data yang dianalisis telah melalui tahapan pembersihan dan transformasi yang sesuai, sehingga dapat menghasilkan visualisasi dan interpretasi yang akurat. Dengan metodologi yang sistematis dan berbasis prinsip ilmiah, diharapkan hasil analisis dapat mencerminkan struktur data yang sebenarnya dan memberikan wawasan yang bermakna terhadap distribusi serta keterpisahan kelas dalam dataset Iris.

a. *Tools* dan Bahasa Pemrograman yang Digunakan

Dalam pelaksanaan analisis ini, digunakan bahasa pemrograman Python karena fleksibilitas dan kelengkapan pustaka (library) untuk keperluan data analysis dan machine learning. Analisis dilakukan sepenuhnya menggunakan ***Google Colaboratory*** **(Colab)**, yang memungkinkan eksekusi kode Python berbasis cloud tanpa instalasi lokal. Adapun *tools* dan *library* yang digunakan:

a) **Python** – Bahasa pemrograman utama untuk seluruh proses analisis.
b) **Pandas** – Untuk membaca dan memproses *dataset*, serta manipulasi *DataFrame*.
c) **NumPy** – Untuk perhitungan numerik, matriks, dan operasi vektor.
d) **Matplotlib** dan **Seaborn** – Untuk visualisasi data dan hasil proyeksi dalam bentuk *scatter plot* dan *boxplot*.
e) **Scikit-learn** – Digunakan khusus untuk *preprocessing* seperti standardisasi (*StandardScaler*) dan penghitungan jarak antar sampel (pdist, squareform).
f) ***Google Colab*** – Platform *notebook* berbasis *cloud* untuk mengembangkan, menjalankan, dan mendokumentasikan kode Python secara interaktif.

b. *Cleaning and Preprocessing Data* Sebelum Implementasi

Sebelum algoritma reduksi dimensi seperti **PCA (*Principal Component Analysis*)** dan **MDS (*Multidimensional Scaling*)** dapat diterapkan, dilakukan serangkaian proses *cleaning* dan *preprocessing* data untuk memastikan bahwa dataset berada dalam kondisi yang optimal dan sesuai dengan prasyarat teknis dari kedua metode tersebut. Proses ini penting karena data yang belum dibersihkan dapat mengandung masalah seperti **nilai yang hilang**, **duplikasi**, atau **skala fitur yang tidak konsisten**, yang dapat memengaruhi keakuratan proyeksi dan menghasilkan interpretasi yang menyesatkan. Oleh karena itu, tahap ini berfungsi sebagai fondasi penting yang menjamin bahwa analisis lanjutan dapat berjalan dengan lancar dan valid.

Dalam analisis ini, beberapa langkah utama dilakukan, mulai dari **memisahkan fitur numerik dari label kategorikal** untuk memfokuskan proses proyeksi hanya pada atribut numerik, **menghapus data duplikat** yang ditemukan dalam dataset, hingga memastikan bahwa tidak terdapat *missing values* atau anomali dalam struktur data. Selanjutnya, dilakukan deteksi terhadap *outlier* menggunakan visualisasi *boxplot*. Meskipun ditemukan beberapa *outlier*, data tersebut tidak dihapus karena dianggap merupakan bagian alami dari variasi biologis dalam dataset *Iris*. Langkah terakhir adalah **standardisasi data menggunakan metode *z-score normalization***, yang bertujuan menyamakan skala seluruh fitur sehingga setiap fitur memiliki pengaruh yang seimbang dalam perhitungan variansi (PCA) maupun jarak antar titik (MDS). Dengan preprocessing yang menyeluruh ini, data menjadi siap untuk direduksi dan divisualisasikan secara efektif dalam ruang berdimensi rendah.

1. *Data Cleaning*

Data cleaning merupakan tahap awal yang sangat penting dalam proses analisis data maupun *machine learning*, karena memastikan bahwa data yang digunakan bebas dari kesalahan, inkonsistensi, dan nilai-nilai yang dapat mengganggu hasil analisis. Proses ini bertujuan untuk meningkatkan kualitas, akurasi, dan keandalan data sehingga dapat memberikan hasil yang lebih valid dan bermakna.

a) *Handling Missing Values*

```
df.isnull().sum()
```

Langkah pertama dalam proses pembersihan data adalah memeriksa keberadaan nilai yang hilang (*missing values*). Pengecekan dilakukan menggunakan fungsi `.isnull().sum()` yang menghitung jumlah nilai kosong pada setiap kolom. Berdasarkan hasil pemeriksaan, tidak ditemukan adanya *missing values* dalam dataset, sehingga tidak diperlukan tindakan lebih lanjut seperti imputasi atau penghapusan baris.

b) *Remove Duplicates*

```
df.duplicated().sum()
```

Selanjutnya, dilakukan pemeriksaan terhadap baris data yang terduplikasi menggunakan fungsi `.duplicated()`. Ditemukan sebanyak **3 baris duplikat** dalam dataset. Duplikasi ini dapat menyebabkan bias dalam analisis karena informasi yang sama dihitung lebih dari sekali. Oleh karena itu, baris-baris duplikat tersebut dihapus menggunakan `.drop_duplicates()` dan indeksnya di-*reset* untuk memastikan struktur data tetap rapi. Hasil akhir adalah baris yang tersisa sejumlah **147**.

c) *Data Validation*

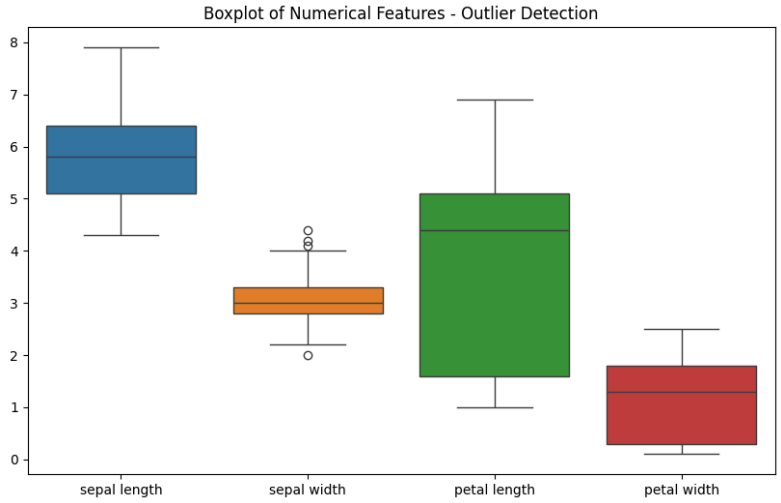
```
df.dtypes

df['class'].unique()
```

Validasi dilakukan untuk memastikan bahwa tipe data pada setiap kolom sesuai dengan yang diharapkan. Kolom fitur seperti *sepal length*, *sepal width*, *petal length*, dan *petal width* dipastikan bertipe numerik, sedangkan kolom label *class* bertipe *string* atau kategorikal. Selain itu, digunakan fungsi `.unique()` untuk

memeriksa bahwa label kelas hanya terdiri dari tiga nilai yang valid, yaitu **Iris-setosa**, **Iris-versicolor**, dan **Iris-virginica**.

d) *Dealing with Outliers*

```
plt.figure(figsize=(10,6))
sns.boxplot(data=df)
plt.title("Boxplot of Numerical Features - Outlier Detection")
plt.show()
```



Gambar 1. Hasil Deteksi *Outlier*

Untuk mendeteksi keberadaan *outlier*, digunakan visualisasi *boxplot* pada setiap fitur numerik. Dari hasil visualisasi, ditemukan beberapa *outlier* khususnya pada fitur **sepal width**. Namun, setelah dipertimbangkan secara seksama, *outlier* tersebut tidak dihapus karena dianggap mencerminkan variasi alami yang valid dalam data biologis, dan dataset *Iris* sendiri merupakan *benchmark dataset* yang telah banyak digunakan dalam literatur. Menghapus *outlier* justru dapat merusak integritas dan representasi data aslinya.

2. *Data Preprocessing*

Data preprocessing merupakan tahap lanjutan yang dilakukan setelah proses pembersihan data selesai. Tahap ini berperan penting dalam

menyiapkan data agar sesuai dengan kebutuhan algoritma *dimensionality reduction* seperti *Principal Component Analysis (PCA)* dan *Multidimensional Scaling (MDS)*, yang dikenal sensitif terhadap skala dan struktur distribusi fitur. Tanpa dilakukan preprocessing yang tepat, hasil proyeksi yang dihasilkan oleh kedua algoritma tersebut dapat menjadi bias atau kurang merepresentasikan pola asli dari data. Oleh karena itu, preprocessing tidak hanya bersifat teknis, tetapi juga strategis dalam memastikan bahwa setiap fitur memberikan kontribusi yang seimbang dalam proses analisis dan visualisasi. Beberapa langkah penting dalam tahap ini meliputi pemisahan antara fitur numerik dan label kategorikal, serta penerapan teknik *z-score normalization* untuk menyamakan skala fitur sebelum digunakan sebagai input dalam PCA dan MDS.

a) Separating Numerical Features and Categorical Label & Standardization (z-score)

Langkah pertama dalam preprocessing adalah memisahkan fitur numerik dari label kategorikal. Empat fitur numerik (*sepal length*, *sepal width*, *petal length*, dan *petal width*) disimpan dalam variabel **X**, sedangkan label kelas **class** disimpan dalam variabel **y**. Hal ini dilakukan karena algoritma PCA dan MDS hanya bekerja pada data numerik, sementara label hanya diperlukan untuk keperluan visualisasi hasil. Setelah fitur numerik dipisahkan, dilakukan proses **standardisasi** menggunakan metode *z-score normalization*. Proses ini memastikan bahwa semua fitur memiliki rata-rata 0 dan standar deviasi 1, sehingga setiap fitur memberikan kontribusi yang seimbang terhadap perhitungan jarak (untuk MDS) maupun variansi (untuk PCA). Standardisasi dilakukan menggunakan *StandardScaler* dari pustaka *scikit-learn*. Statistik sebelum standardisasi menunjukkan bahwa fitur-fitur memiliki skala dan sebaran yang berbeda, yang dapat mempengaruhi hasil analisis. Namun setelah distandarisasi, semua fitur berada dalam skala yang seragam, seperti yang ditunjukkan oleh nilai *mean* mendekati nol dan *standard*

deviation mendekati satu pada setiap fitur. Dengan demikian, hasil proyeksi dari PCA dan MDS menjadi lebih stabil dan representatif.

```
# Separate features (X) and label (y)
X = df.drop(columns=['class'])  # Numerical
features only
y = df['class']                 # Label column

# Standardize features using z-score
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Display statistics before and after
standardization
print("Statistics before standardization:")
print(pd.DataFrame(X,
columns=X.columns).describe())

print("\nStatistics after standardization:")
print(pd.DataFrame(X_scaled,
columns=X.columns).describe())
```

```
Statistics before standardization:
       sepal length  sepal width  petal length  petal width
count    147.000000   147.000000    147.000000   147.000000
mean       5.856463     3.055782      3.780272     1.208844
std        0.829100     0.437009      1.759111     0.757874
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.400000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000

Statistics after standardization:
       sepal length   sepal width  petal length   petal width
count  1.470000e+02  1.470000e+02  1.470000e+02  1.470000e+02
mean  -4.833624e-17  1.691768e-16 -2.416812e-16 -3.383537e-16
std    1.003419e+00  1.003419e+00  1.003419e+00  1.003419e+00
min   -1.883710e+00 -2.424189e+00 -1.585902e+00 -1.468099e+00
25%   -9.155095e-01 -5.873036e-01 -1.243654e+00 -1.203301e+00
50%   -6.833389e-02 -1.280822e-01  3.535005e-01  1.206904e-01
75%    6.578166e-01  5.607500e-01  7.527893e-01  7.826860e-01
max    2.473193e+00  3.086468e+00  1.779532e+00  1.709480e+00
```

Gambar 2. Hasil Standardisasi

c. *Langkah-Langkah Implementasi PCA dan MDS*

Setelah data melalui tahap *cleaning* dan *preprocessing* yang menyeluruh, langkah berikutnya adalah menerapkan teknik **reduksi dimensi** sebagai bagian dari

proses ***Dimensionality Reduction / Feature Engineering***. Tahapan ini bertujuan untuk menyederhanakan struktur data tanpa kehilangan informasi penting yang terkandung di dalamnya. Dengan mengurangi jumlah dimensi, proses analisis menjadi lebih efisien, visualisasi menjadi lebih informatif, serta risiko terhadap *curse of dimensionality* dapat diminimalkan. Dalam proyek ini, dua metode populer digunakan, yaitu ***Principal Component Analysis* (PCA)** dan ***Multidimensional Scaling* (MDS)**. Kedua teknik ini memiliki pendekatan yang berbeda, namun sama-sama digunakan untuk memproyeksikan data berdimensi tinggi ke dalam ruang dua dimensi agar pola, keterpisahan antar kelas, dan struktur internal data lebih mudah dipahami. Berikut adalah penjabaran mengenai **langkah-langkah implementasi PCA dan MDS** yang dilakukan dalam analisis ini.

1. ***Principal Component Analysis (PCA)***

   ***Principal Component Analysis* (PCA)** adalah salah satu metode *dimensionality reduction* yang bertujuan untuk mengubah data berdimensi tinggi menjadi data berdimensi lebih rendah dengan tetap mempertahankan sebanyak mungkin informasi (variansi) dari data asli. Berikut adalah tahapan implementasi PCA dari awal (*from scratch*) pada dataset *Iris*.

   a) Menghitung Rata-Rata Setiap Fitur

   Langkah pertama dalam PCA adalah menghitung rata-rata (*mean*) dari masing-masing fitur. Tujuannya adalah untuk *centering* data, yakni menggeser distribusi data sehingga memiliki rata-rata nol. Ini merupakan langkah dasar untuk membentuk matriks kovarians pada tahap selanjutnya.

```
# Step 1: Calculate the mean of each feature
mean_vec = np.mean(X_scaled, axis=0)
print("Mean vector:\n", mean_vec)
```

   Output dari perhitungan ini menunjukkan bahwa seluruh fitur memiliki nilai rata-rata sangat dekat dengan nol karena sebelumnya telah distandarisasi menggunakan *z-score*.

   b) Menghitung Matriks Kovarians

   Setelah data dicentering, langkah berikutnya adalah menghitung **matriks kovarians** antar fitur. Matriks ini menunjukkan sejauh mana fitur-fitur dalam data berkorelasi satu sama lain.

```
# Step 2: Compute the covariance matrix (between features)
cov_matrix = np.cov(X_scaled.T)
print("Matriks kovarians:\n", cov_matrix)
```

Matriks kovarians bersifat simetris, dan nilai pada diagonal utamanya bernilai mendekati 1 karena data sudah distandarisasi. Nilai di luar diagonal menggambarkan hubungan antar fitur.

c) Melakukan Dekomposisi Eigen

PCA menggunakan **dekomposisi eigen** terhadap matriks kovarians untuk menentukan arah variansi terbesar (komponen utama). Hasil dari proses ini adalah *eigenvalues* dan *eigenvectors*.

```
# Step 3: Perform eigen decomposition (eigenvalues & eigenvectors)
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)
```

Eigenvalues menunjukkan seberapa besar varian yang dapat dijelaskan oleh setiap komponen, sedangkan *eigenvectors* adalah arah dari komponen tersebut dalam ruang fitur.

d) Mengurutkan Komponen Utama Berdasarkan Nilai Eigen

Komponen utama diurutkan berdasarkan nilai *eigenvalue* dari yang terbesar ke terkecil. Dua komponen teratas dipilih untuk proyeksi ke ruang dua dimensi karena menjelaskan proporsi varian terbesar dalam data.

```
# Step 4: Sort components by descending eigenvalues
sorted_idx = np.argsort(eigenvalues)[::-1]
eigenvalues = eigenvalues[sorted_idx]
eigenvectors = eigenvectors[:, sorted_idx]

print("Eigenvalues:\n", eigenvalues)
print("\nEigenvectors:\n", eigenvectors)
```

Hasil menunjukkan bahwa komponen pertama memiliki *eigenvalue* tertinggi (sekitar 2.93), yang berarti menjelaskan sebagian besar varian dalam data.

e) Memilih Dua Komponen Utama

Setelah pengurutan, dua vektor eigen dengan nilai tertinggi dipilih

untuk membentuk **matriks proyeksi 2D**. Komponen ini menjadi sumbu baru dalam ruang berdimensi rendah.

```
# Step 5: Select the top two principal components
W = eigenvectors[:, :2]  # 2D projection matrix
```

Matriks ini akan digunakan untuk mentransformasikan data asli ke dimensi dua.

f) Memproyeksikan Data ke Ruang 2D

Langkah terakhir adalah melakukan proyeksi data ke ruang 2D menggunakan matriks komponen utama yang telah dipilih. Hasil proyeksi inilah yang akan divisualisasikan.

```
# Step 6: Project the data
X_pca = X_scaled @ W
print("Data dimension after PCA:", X_pca.shape)
```

Output menunjukkan bahwa data telah berhasil direduksi menjadi dimensi 2, dengan bentuk array berukuran $(\mathbf{147}, 2)$.

2. *Multidimensional Scaling (MDS)*

Multidimensional Scaling **(MDS)** adalah salah satu metode *dimensionality reduction* yang berfokus pada pelestarian **jarak antar sampel** dari ruang berdimensi tinggi ke dimensi yang lebih rendah. Tidak seperti PCA yang berbasis pada variansi, MDS bekerja langsung dari matriks jarak, sehingga mampu menangkap pola distribusi spasial data berdasarkan kemiripan (*similarity*) antar observasi. Berikut adalah tahapan implementasi MDS dari awal pada dataset *Iris*:

a) Menghitung Jarak Euclidean dan Membentuk Matriks Jarak

Langkah pertama adalah menghitung seluruh jarak Euclidean antar baris data (sampel), lalu membentuk sebuah **matriks jarak kuadrat berdimensi $n \times n$**. Setiap elemen dalam matriks ini menyatakan jarak antara dua sampel.

```
# Step 1 & 2: Compute pairwise Euclidean distances
and build distance matrix
D = squareform(pdist(X_scaled, metric='euclidean'))
# D is a (n x n) distance matrix
```

```
D[:5, :5]
```

Output yang dihasilkan adalah matriks simetris dengan diagonal bernilai nol (jarak dengan dirinya sendiri). Matriks ini menjadi dasar utama dalam transformasi MDS.

b) Mengaplikasikan Algoritma MDS Klasik (*Classical* MDS)

Setelah mendapatkan matriks jarak, dilakukan ***double centering*** untuk mengubah matriks jarak menjadi ***Gram matrix*** (matriks produk dalam), yang memungkinkan dapat melakukan dekomposisi eigen terhadapnya.

```
# Number of samples
n = D.shape[0]

# Centering matrix
H = np.eye(n) - np.ones((n, n)) / n

# Double centering to get the Gram matrix (inner
product matrix)
B = -0.5 * H @ (D ** 2) @ H
```

Transformasi ini mengubah informasi spasial dari matriks jarak menjadi bentuk matriks simetris yang dapat diurai secara matematis seperti pada PCA.

c) Melakukan Dekomposisi Eigen

Dekomposisi eigen dilakukan terhadap ***Gram matrix*** untuk mendapatkan *eigenvalues* dan *eigenvectors*. Nilai eigen terbesar mewakili dimensi yang paling menjelaskan struktur jarak antar titik.

```
# Eigen decomposition
eigenvalues_mds, eigenvectors_mds =
np.linalg.eigh(B)

# Sort by largest eigenvalues
idx = np.argsort(eigenvalues_mds)[::-1]
eigenvalues_mds = eigenvalues_mds[idx]
eigenvectors_mds = eigenvectors_mds[:, idx]
```

Urutan komponen utama ditentukan berdasarkan *eigenvalues* dari yang terbesar ke terkecil, mirip seperti pada PCA.

c) Mengambil Dua Dimensi Teratas sebagai Proyeksi

Dua komponen utama yang memiliki *eigenvalue* terbesar diambil untuk membentuk representasi data dalam ruang dua dimensi. Setiap titik hasil proyeksi dihitung dari vektor eigen dikalikan akar dari nilai eigennya.

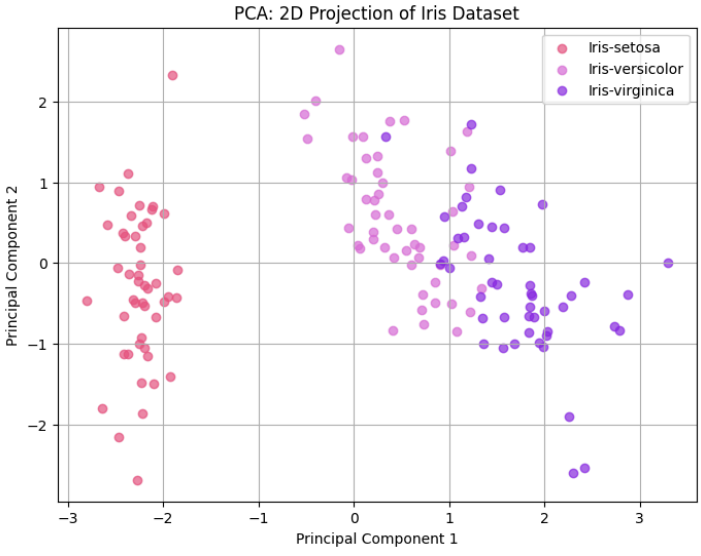
```
# Take the top 2 dimensions
X_mds = eigenvectors_mds[:, :2] *
np.sqrt(eigenvalues_mds[:2])
print("Data dimension after MDS:", X_mds.shape)
```

Hasilnya adalah array berukuran (**147**, 2) yang menunjukkan koordinat setiap data dalam ruang MDS 2D.

3. Hasil Visualisasi

Gambar di bawah ini menunjukkan **hasil visualisasi dua dimensi dari dataset Iris yang telah direduksi menggunakan dua teknik reduksi dimensi**, yaitu *Principal Component Analysis* (PCA) dan *Multidimensional Scaling* (MDS). Warna digunakan untuk membedakan tiga kelas bunga iris: Iris-setosa (pink), Iris-versicolor (ungu muda), dan Iris-virginica (biru violet).

a. *Principal Component Analysis* (PCA)



Gambar 3. Hasil Visualisasi PCA

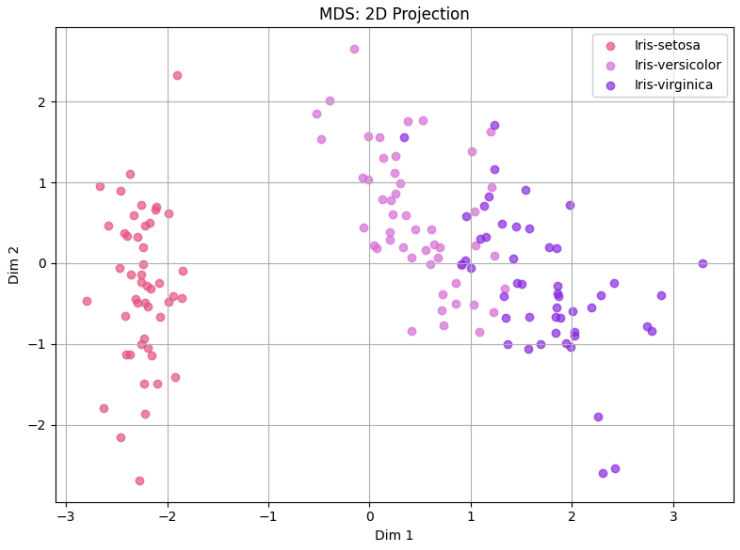
Visualisasi hasil proyeksi dua dimensi menggunakan ***Principal Component Analysis* (PCA)** memperlihatkan bahwa kelas **Iris-setosa** terpisah dengan sangat

jelas dari dua kelas lainnya. Hal ini menunjukkan bahwa kelas tersebut **dapat dipisahkan secara linier** dalam ruang fitur asli. Sementara itu, kelas **Iris-versicolor** dan **Iris-virginica** tampak memiliki area yang saling tumpang tindih, yang mengindikasikan bahwa distribusi fitur dari kedua kelas tersebut cukup mirip sehingga lebih sulit untuk dibedakan hanya dengan menggunakan dua komponen utama.

Komponen utama pertama (*Principal Component 1*) merupakan komponen yang menangkap variansi terbesar dalam data, dan kemungkinan besar berasosiasi dengan fitur **panjang dan lebar kelopak** (*petal length* dan *petal width*), yaitu dua fitur yang secara umum dikenal sangat membedakan ketiga jenis bunga dalam dataset Iris. Walaupun PCA mampu mereduksi dimensi menjadi dua komponen yang dapat divisualisasikan, sifatnya yang **linier** membuatnya **kurang mampu menangkap hubungan non-linier** antar kelas, khususnya antara Iris-versicolor dan Iris-virginica.

Meskipun demikian, hasil visualisasi ini tetap berguna sebagai alat eksplorasi awal. Visualisasi ini menunjukkan bahwa PCA dapat mereduksi dimensi data tanpa menghilangkan struktur kelas yang signifikan, terutama pada kelas yang paling terpisah seperti **Iris-setosa**.

b. *Multidimensional Scaling* (MDS)



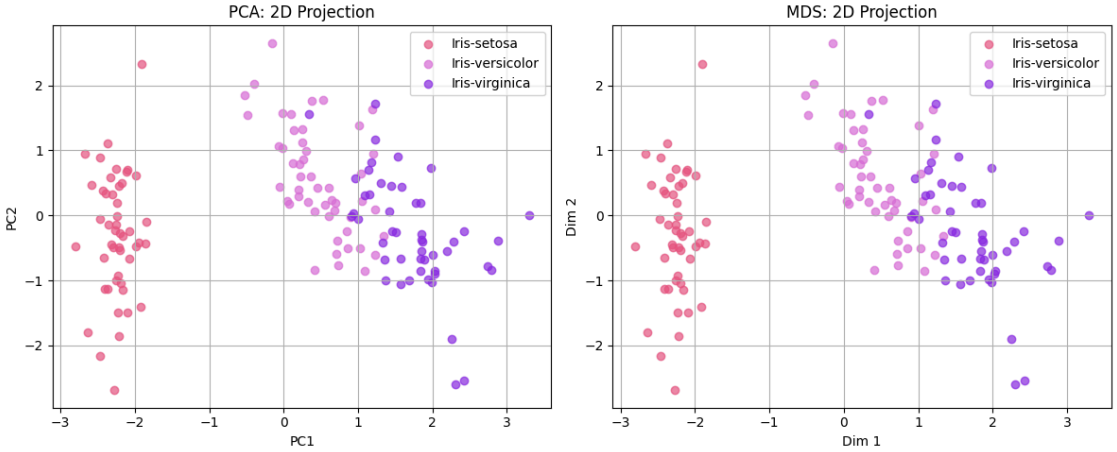
Gambar 4. Hasil Visualisasi MDS

Visualisasi hasil proyeksi dua dimensi menggunakan ***Multidimensional Scaling*** **(MDS)** juga menunjukkan keterpisahan yang sangat jelas dari kelas **Iris-setosa** terhadap dua kelas lainnya. Hal ini mengindikasikan bahwa hubungan berbasis jarak antar sampel pada ruang fitur asli memang secara alami memisahkan kelompok ini dari yang lain. Namun, distribusi data untuk kelas **Iris-versicolor** dan **Iris-virginica** tampak **lebih saling bercampur (*intermixed*)**, dengan beberapa titik berada di area transisi antar keduanya. Pola ini menunjukkan bahwa, berdasarkan **jarak antar pasangan sampel (*pairwise distances*)**, perbedaan antara dua spesies ini tidak terlalu mencolok.

Berbeda dengan PCA yang mendasarkan proyeksi pada **arah variansi terbesar**, MDS fokus pada upaya mempertahankan **jarak relatif antar titik data**. Pendekatan ini memungkinkan MDS untuk lebih sensitif terhadap **struktur non-linier** dalam data. Dalam konteks ini, posisi titik-titik dalam ruang MDS merepresentasikan hubungan kedekatan antar sampel dalam ruang berdimensi tinggi, meskipun tidak menghasilkan sumbu yang dapat ditafsirkan langsung terhadap fitur-fitur aslinya.

Secara keseluruhan, visualisasi MDS memberikan perspektif yang **komplementer terhadap PCA**, dengan menekankan bagaimana jarak antar data mencerminkan kedekatan kelas. Meskipun dalam dataset ini MDS tidak memberikan keterpisahan antar kelas yang lebih baik dibandingkan PCA, metode ini tetap memperkuat posisi terpisah dari **Iris-setosa** dan menyoroti **ambiguitas antara kelas Versicolor dan Virginica**.

4. Analisis Perbandingan



Gambar 5. Analisis Perbandingan PCA dan MDS

a. Keterpisahan Antar Kelas

Hasil proyeksi dua dimensi menggunakan **PCA** dan **MDS** sama-sama menunjukkan bahwa kelas **Iris-setosa** terpisah dengan sangat jelas dari dua kelas lainnya. Hal ini menegaskan bahwa kelas tersebut memiliki karakteristik fitur yang paling berbeda, dan baik PCA maupun MDS mampu menangkap perbedaan tersebut secara konsisten.

Namun, untuk dua kelas lainnya yaitu **Iris-versicolor** dan **Iris-virginica**, kedua metode menunjukkan adanya **tumpang tindih (*overlap*)**. Meskipun begitu, PCA terlihat memberikan pemisahan yang sedikit lebih tegas antara dua kelas tersebut dibandingkan dengan MDS, meskipun tidak signifikan. Pada MDS, titik-titik dari kedua kelas ini lebih banyak tersebar dan saling berdekatan, menunjukkan bahwa dalam ruang jarak antar titik, perbedaan antar kelas ini memang tidak terlalu besar.

b. Interpretasi Dimensi Hasil Reduksi

Pada PCA, dua dimensi hasil proyeksi adalah **kombinasi linier dari fitur-fitur asli**, yang disusun berdasarkan **kontribusi variansi terbesar**. Dimensi pertama (PC1) memuat informasi paling banyak, diikuti oleh dimensi kedua (PC2). Karena dibentuk secara linier dari fitur asli, komponen-komponen PCA **masih**

dapat diinterpretasikan secara matematis dan berguna untuk pemahaman struktural data.

Sementara itu, MDS tidak membentuk dimensi dari kombinasi fitur asli, melainkan dari proses dekomposisi matriks jarak antar sampel. Oleh karena itu, dimensi hasil MDS **tidak dapat diinterpretasikan secara langsung** karena tidak mewakili fitur atau atribut tertentu. Tujuan utama MDS adalah untuk **mempertahankan hubungan jarak** antar sampel dalam ruang proyeksi berdimensi rendah.

c. Kelebihan dan Keterbatasan Masing-Masing Metode

Perbandingan **kelebihan dan keterbatasan masing-masing metode** disajikan pada tabel berikut berdasarkan beberapa aspek utama yang relevan dalam analisis reduksi dimensi.

Aspek	PCA	MDS
Dasar Metode	Variansi antar fitur	Jarak antar sampel
Interpretabilitas Dimensi	Tinggi (berbasis fitur asli)	Rendah (berbasis posisi spasial)
Efisiensi Komputasi	Cepat, sangat efisien	Lebih lambat, apalagi untuk data besar
Kemampuan Tangkap Pola	Baik untuk hubungan linier	Lebih fleksibel, bisa tangkap struktur non-linier
Kesesuaian untuk Visualisasi	Sangat baik jika data memiliki fitur dengan variansi tinggi	Baik jika ingin eksplorasi kemiripan atau kedekatan antar data

Meskipun **PCA dan MDS berasal dari pendekatan matematis yang berbeda**, hasil visualisasi proyeksi dua dimensi pada dataset Iris menunjukkan pola distribusi yang **sangat mirip**. Hal ini disebabkan oleh karakteristik dataset yang **bersih, berdimensi rendah, dan memiliki struktur linier yang kuat**. Oleh karena itu, baik proyeksi berbasis variansi (PCA) maupun berbasis jarak (MDS) menghasilkan representasi kelas yang hampir identik.

Namun, pada dataset yang lebih kompleks atau tidak terstruktur secara linier, kedua metode ini dapat menghasilkan proyeksi yang **sangat berbeda**, dan pemilihan metode yang tepat menjadi penting tergantung pada tujuan analisis. Secara umum, **PCA lebih unggul**

dalam hal efisiensi dan interpretasi, sedangkan **MDS lebih fleksibel dalam menangkap struktur non-linier dan pola jarak antar data**.

5. Kesimpulan dan Refleksi

Hasil analisis dan visualisasi pada dataset *Iris* menunjukkan bahwa baik **Principal Component Analysis (PCA)** maupun **Multidimensional Scaling (MDS)** sama-sama mampu merepresentasikan struktur kelas dalam bentuk dua dimensi dengan cukup baik. Kedua metode berhasil menunjukkan keterpisahan yang jelas pada kelas **Iris-setosa**, serta area tumpang tindih antara **Iris-versicolor** dan **Iris-virginica**, yang mengindikasikan kemiripan distribusi fitur antara dua kelas tersebut.

Dari sisi metode, PCA memberikan dimensi hasil proyeksi yang masih dapat diinterpretasikan secara matematis karena dibentuk dari kombinasi linier fitur asli dan diurutkan berdasarkan kontribusi variansi. Sementara itu, MDS tidak menghasilkan dimensi yang terikat pada fitur, melainkan lebih menekankan pada pelestarian hubungan spasial atau jarak antar sampel. Meskipun secara prinsip berbeda, hasil visualisasi keduanya tampak sangat mirip dalam kasus ini, yang mencerminkan karakteristik dataset *Iris* yang terstruktur dengan baik dan bersifat linier.

Melalui implementasi algoritma dari awal (*from scratch*), pemahaman terhadap konsep dasar PCA dan MDS menjadi lebih mendalam, khususnya mengenai bagaimana setiap langkah—dari standardisasi data, pembentukan matriks kovarians atau matriks jarak, hingga dekomposisi eigen—berkontribusi pada proses reduksi dimensi. Refleksi dari hasil ini menunjukkan bahwa pemilihan metode reduksi dimensi perlu disesuaikan dengan sifat data dan tujuan analisis, serta bahwa kombinasi pendekatan eksploratif seperti PCA dan MDS dapat memberikan sudut pandang yang saling melengkapi terhadap struktur data berdimensi tinggi.

6. Referensi

Books:

- Jolliffe, Ian T. *Principal Component Analysis*. 2nd ed. Springer Series in Statistics. New York: Springer, 2002.

- Borg, Ingwer, and Patrick J. F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. 2nd ed. New York: Springer, 2005.

Journal Articles and Preprints:

- van der Maaten, Laurens, and Geoffrey Hinton. "Visualizing Data Using t-SNE." *Journal of Machine Learning Research* 9 (2008): 2579–2605.

- McInnes, Leland, John Healy, and James Melville. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction." *arXiv preprint* arXiv:1802.03426 (2018). https://arxiv.org/abs/1802.03426.

Web Documentation and Libraries:

- Scikit-learn. "sklearn.decomposition.PCA." Accessed March 27, 2025. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

- Scikit-learn. "sklearn.preprocessing.StandardScaler." Accessed March 27, 2025. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

- Scikit-learn. "sklearn.datasets.load_iris." Accessed March 27, 2025. https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html.

- NumPy. "Linear Algebra (numpy.linalg)." Accessed March 27, 2025. https://numpy.org/doc/stable/reference/routines.linalg.html.

- SciPy. "scipy.spatial.distance.pdist." Accessed March 27, 2025. https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.pdist.html.

- Matplotlib. "Matplotlib Pyplot." Accessed March 27, 2025. https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html.

- pandas. "API Reference." Accessed March 27, 2025. https://pandas.pydata.org/docs/reference/index.html.

7. Lampiran

Lampiran Link ***Google Colaboratory*** **Tugas PCA dan MDS**:

18222023_Thalita Zahra Sutejo_Tugas_PCA_MDS.ipynb

linked: *https://colab.research.google.com/drive/1gUe9PwsLdgskNYFFg1pzmCxdEf6YeLMf?authuser=0#scrollTo=uBv_ZDM7HR2B*

Lampiran Link ***Github*** **Tugas PCA dan MDS**:

https://**github.com/thalitazhrr/PCAandMDS_Task**