Principal Component Analysis(PCA)

PCA adalah teknik unsupervised learning yang digunakan untuk mereduksi dimensi data. PCA bertujuan untuk menemukan kombinasi linear dari fitur-fitur asli yang menangkap sebagian besar varians dalam data. Kombinasi linear ini disebut principal components (Komponen Utama).

# Mengapa Menggunakan PCA?

* **Mengurangi Dimensi:** PCA dapat mengurangi jumlah fitur dalam dataset, yang dapat mempercepat komputasi dan mengurangi risiko overfitting.
* **Visualisasi:** PCA dapat digunakan untuk memvisualisasikan data berdimensi tinggi dalam ruang dua atau tiga dimensi.
* **Mengidentifikasi Pola:** PCA dapat membantu mengidentifikasi pola-pola dalam data yang mungkin tidak terlihat dalam ruang fitur asli.

# Cara Kerja PCA:

1. **Standardisasi Data:** PCA sensitif terhadap skala data, sehingga fitur-fitur perlu distandarisasi (rata-rata = 0, standar deviasi = 1) sebelum menerapkan PCA.
2. **Menghitung Covariance Matrix:** Matriks kovariansi mengukur hubungan linear antara setiap pasangan fitur.
3. **Menghitung Eigenvalues dan Eigenvectors:** Eigenvalues dan Eigenvectors dari matriks kovariasi memberikan informasi tentang varians yang dijelaskan oleh setiap komponen utama dan arah dari komponen utama tersebut.
4. **Memilih Komponen Utama:** Kita dapat memilih beberapa komponen utama pertama yang menjelaskan sebagian varians dalam data.

# Penerapan PCA dengan Scikit-Learn

Scikit-learn menyediakan kelas PCA untuk melakukan PCA.

**Tugas Hari 28:**

Menggunakan Dataset “Iris”

**Tugas:**

1. **Membuat Model PCA:**

* Gunakan PCA dari scikit-learn untuk membuat model PCA dengan jumlah komponen utama = 2.
* Terapkan PCA pada dataset Iris yang sudah distandarisasi.

1. **Visualisasi:**

* Buat scatter plot dari data yang sudah ditransformasi dengan PCA, dengan warna titik berbeda untuk setiap jenis bunga iris.
* Beri label sumbu x dan y dengan “Principal Component 1” dan “Principal Component 2”.

**Contoh Kode (Scikit-Learn dan Matplotlib)**

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

# ... (kode untuk membaca dan mempersiapkan dataset Iris)

# Standardisasi data

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# PCA

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X\_scaled)

# Visualisasi

plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=y, cmap='viridis')

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title('PCA dari Dataset Iris')

plt.show()

**Catatan:**

Coba Variasikan jumlah komponen utama (misalnya, 3 atau 4) dan lihat bagaimana pengaruhnya pada hasil visualisasi.

**Selamat Mengerjakan Tugas! 🙂**

**Tugas:**

1. **Membuat Model PCA:**

**from sklearn.decomposition import PCA**

**from sklearn.preprocessing import StandardScaler**

**# ... (Kode untuk Membaca dan Mempersiapkan dataset Iris)**

**# Standardisasi data**

**scaler = StandardScaler()**

**X\_scale = scaler.fit\_transform(X)**

**# PCA**

**pca = PCA(n\_components=2)**

**X\_pca = pca.fit\_transform(X\_scale)**

1. **Visualisasi:**

**# Visualisasi**

**plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=y, cmap='viridis')**

**plt.xlabel('Principal Component 1')**

**plt.ylabel('Principal Component 2')**

**plt.title('PCA dari Dataset Iris')**

**plt.show()**

## **Modifikasi Kode untuk Variasi Jumlah Komponen Utama:**

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

# ... (kode untuk membaca dan mempersiapkan dataset Iris)

# Standardisasi data

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Coba beberapa jumlah komponen utama

for n\_components in [2, 3]: # Anda dapat mengubah nilai ini sesuai keinginan

# PCA

pca = PCA(n\_components=n\_components)

X\_pca = pca.fit\_transform(X\_scaled)

# Visualisasi

plt.figure(figsize=(8, 6))

plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=y, cmap='viridis')

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2' if n\_components == 2 else 'Principal Component 3')

plt.title(f'PCA Iris Dataset ({n\_components} komponen utama)')

plt.show()

**Penjelasan Perubahan:**

1. **Looping:** Kita menggunakan loop for untuk mencoba beberapa nilai n\_components (jumlah komponen utama), misalnya 2 dan 3. Anda dapat mengubah nilai-nilai ini sesuai kebutuhan.
2. **Label Sumbu Y:** Kita menggunakan kondisi if n\_components == 2 else 'Principal Component 3' untuk mengatur label sumbu y dengan benar, tergantung pada jumlah komponen utama yang digunakan.
3. **Judul Plot:** Kita menambahkan informasi tentang jumlah komponen utama yang digunakan pada judul plot.

**Interpretasi Hasil:**

Setelah menjalankan kode ini, Anda akan melihat dua scatter plot:

* **2 Komponen Utama:** Menampilkan data Iris dalam dua dimensi setelah diproyeksikan ke dua komponen utama pertama.
* **3 Komponen Utama:** Menampilkan data Iris dalam tiga dimensi setelah diproyeksikan ke tiga komponen utama pertama (Anda akan mendapatkan plot 3D).

**Perbandingan:**

Amati kedua plot tersebut dan perhatikan perbedaannya. Ketika Anda menggunakan lebih banyak komponen utama, Anda akan mendapatkan visualisasi yang lebih informatif karena lebih banyak varians dalam data asli yang tertangkap. Namun, visualisasi juga menjadi lebih sulit untuk diinterpretasikan jika jumlah komponen utama terlalu banyak.

**Kesimpulan:**

Dengan mengubah jumlah komponen utama, Anda dapat melihat bagaimana hal tersebut mempengaruhi hasil visualisasi PCA. Ini membantu Anda memilih jumlah komponen utama yang optimal untuk analisis atau visualisasi data Anda.

**Catatan:**

* Jika Anda ingin mencoba lebih dari 3 komponen utama, Anda perlu menggunakan library visualisasi 3D seperti mpl\_toolkits.mplot3d dari Matplotlib.
* Anda juga dapat menggunakan metode seperti scree plot atau explained variance ratio untuk membantu Anda memilih jumlah komponen utama yang optimal.