Principal Component Analysis (PCA) - Teori dan Penerapan

# Konsep PCA

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

# Tujuan PCA

PCA memiliki beberapa tujuan utama;

1. **Reduksi Dimensi:** Mengurangi jumlah fitur dalam dataset, yang dapat mempercepat komputasi, mengurangi kompleksitas model, dan menghindari overfitting.
2. **Visualisasi:** Memvisualisasikan data berdimensi tinggi dalam ruang dua atau tiga dimensi, sehingga lebih mudah untuk memahami pola-pola dalam data.
3. **Identifikasi Fitur Penting:** Mengidentifikasi fitur-fitur yang paling berkontribusi terhadap varians data, yang dapat membantu dalam interpretasi dan pemilihan fitur.

# 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. **Hitung Matriks Kovariansi:** Matriks kovariansi mengukur variansi dan korelasi antara setiap pasangan fitur.
3. **Hitung Eigenvalues dan Eigenvector:** Eigenvalues dan Eigenvectors dari matriks kovariansi memberikan informasi tentang varians yang dijelaskan oleh setiap komponen utama dan arah dari komponen utama tersebut.
4. **Pilih Komponen Utama:** Pilih beberapa komponen utama yang menjelaskan sebagian besar varians dalam data. Jumlah komponen utama yang dipilih tergantung pada seberapa banyak varians yang ingin Anda pertahankan dan seberapa banyak reduksi dimensi yang Anda inginkan.

**Implementasi PCA dengan Scikit-Learn**

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

# Standardisasi data

scaler = StandardScaler()

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

# PCA

pca = PCA(n\_components=2) # Mengubah jumlah komponen utama di sini

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

**Visualisasi Hasil PCA**

# Plot

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 of Iris Dataset')

plt.show()

**Tugas Hari 35:**

1. **Memilih Jumlah Komponen Utama:** Coba variasikan jumlah komponen utama yang digunakan dalam PCA pada dataset Iris (Misalnya 2, 3, atau 4). Amati bagaimana hal ini mempengaruhi hasil visualisasi dan interaksi Anda.
2. **Explained Variance Ratio:** Gunakan atribut explained\_variance\_ratio\_ dari objek PCA untuk melihat proporsi varians yang dijelaskan oleh setiap komponen utama.
3. **Scree Plot:** Buat scree plot untuk memvisualisasikan explained variance ratio dari setiap komponen utama. Scree plot dapat membantu Anda menentukan jumlah komponen utama yang optimal.

**Contoh Kode (Scree Plot)**

plt.plot(np.cumsum(pca.explained\_variance\_ratio\_))

plt.xlabel('Number of components')

plt.ylabel('Cumulative explained variance')

plt.show()

**Selamat Mengerjakan Tugas! 🙂**

**Tugas:**

**from sklearn.decomposition import PCA**

**from sklearn.preprocessing import StandardScaler**

**import matplotlib.pyplot as plt**

**# Standardisasi Data**

**scaler = StandardScaler()**

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

**# PCA**

**pca = PCA(n\_components=2) # Mengubah JUmlah Komponen Utama disini**

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

**# Plot**

**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 of Iris Dataset')**

**plt.show()**

**plt.plot(np.cumsum(pca.explained\_variance\_ratio\_))**

**plt.xlabel('Number of components')**

**plt.ylabel('Cumulative explained variance')**

**plt.show()**

**Penjelasan Kode:**

1. **Standardisasi Data:**
   * scaler = StandardScaler(): Membuat objek StandardScaler.
   * X\_scaled = scaler.fit\_transform(X): Melakukan standardisasi fitur-fitur pada dataset Iris (X). Ini penting karena PCA sensitif terhadap skala fitur.
2. **PCA:**
   * pca = PCA(n\_components=2): Membuat objek PCA dengan 2 komponen utama. Anda dapat mengubah nilai n\_components untuk mencoba variasi jumlah komponen utama.
   * X\_pca = pca.fit\_transform(X\_scaled): Menerapkan PCA pada data yang sudah distandarisasi. X\_pca adalah hasil transformasi PCA yang berisi koordinat data pada sumbu-sumbu komponen utama baru.
3. **Visualisasi (Scatter Plot):**
   * plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=y, cmap='viridis'): Membuat scatter plot dengan:
     + Sumbu x: Principal Component 1 (PC1)
     + Sumbu y: Principal Component 2 (PC2)
     + Warna titik (c): Berdasarkan label kelas bunga iris (y)
     + Colormap (cmap): 'viridis'
4. **Visualisasi (Scree Plot):**
   * plt.plot(np.cumsum(pca.explained\_variance\_ratio\_)): Membuat scree plot dengan:
     + Sumbu x: Jumlah komponen utama
     + Sumbu y: Cumulative explained variance (varians kumulatif yang dijelaskan)

**Interpretasi Hasil:**

* **Scatter Plot:**
  + Plot ini menunjukkan bagaimana tiga jenis bunga iris (Setosa, Versicolor, Virginica) terpisah dalam ruang dua dimensi yang dibentuk oleh dua komponen utama pertama.
  + Jika titik-titik data dari kelas yang berbeda terpisah dengan baik, maka PCA berhasil menemukan kombinasi fitur yang baik untuk membedakan kelas-kelas tersebut.
* **Scree Plot:**
  + Plot ini menunjukkan proporsi varians kumulatif yang dijelaskan oleh setiap komponen utama.
  + Anda dapat menggunakan plot ini untuk menentukan berapa banyak komponen utama yang cukup untuk menangkap sebagian besar varians dalam data. Biasanya, kita akan memilih titik "siku" (elbow point) pada plot sebagai jumlah komponen utama yang optimal.