TUGAS 6 Prak ML

- Nama : Nashirudin Baqiy
- NIM : 24060119130045
- Lab : A2
- Dataset : Air Quality

A. Principal Component Analysis

Principal Component Analysis adalah teknik untuk ekstraksi fitur - jadi ini menggabungkan variabel masukan kita dengan cara tertentu, lalu kita dapat menghilangkan variabel "paling tidak penting" sambil tetap mempertahankan bagian paling berharga dari semua variabel! Sebagai manfaat tambahan, masingmasing variabel "baru" setelah PCA semuanya tidak bergantung satu sama lain. Ini menguntungkan karena asumsi model linier mengharuskan variabel independen kami independen satu sama lain. Jika kita memutuskan untuk menyesuaikan model regresi linier dengan variabel "baru" ini (lihat "regresi komponen utama" di bawah), asumsi ini akan terpenuhi. PCA dapat digunakan saat:

1. Ketika Anda ingin mengurangi jumlah variabel, tetapi tidak dapat mengidentifikasi variabel untuk sepenuhnya dihapus,
2. Ketika Anda ingin memastikan variabel Anda tidak bergantung satu sama lain
3. Ketika variabel independen Anda kurang dapat ditafsirkan

B. Cara Kerja PCA

1. Hitung matriks yang merangkum bagaimana semua variabel kita berhubungan satu sama lain.
2. Pecah matriks ini menjadi dua komponen terpisah: arah dan besaran.
3. Ubah data asli agar selaras dengan petunjuk ini

C. Algoritma Metode Principal Component Analysis

Sebelum memulai, Anda harus mengatur data tabel dengan n baris dan kemungkinan p+1 kolom, di mana satu kolom sesuai dengan variabel dependen Anda (biasanya dilambangkan Y) dan kolom p di mana masing-masing sesuai dengan variabel independen (matriks yang biasanya dilambangkan X).

1. Jika variabel Y ada dan merupakan bagian dari data Anda, maka pisahkan data Anda menjadi Y dan X, seperti yang didefinisikan di atas. (Catatan: jika tidak ada kolom untuk Y, tidak apa-apa - lompat ke poin selanjutnya!)

2. Ambil matriks variabel bebas X dan, untuk setiap kolom, kurangi rata-rata kolom tersebut dari setiap entri. (Ini memastikan bahwa setiap kolom memiliki rata-rata nol.)
3. Putuskan apakah akan melakukan standarisasi atau tidak. Panggil matriks Z yang berpusat (dan mungkin terstandarisasi).
4. Ambil matriks Z, ubah urutannya, dan kalikan matriks yang ditransposekan dengan Z. (secara matematis, kita akan menuliskannya sebagai Z^TZ.) Matriks yang dihasilkan adalah matriks kovariansi Z, hingga sebuah konstanta.
5. Hitung vektor eigen dan nilai eigen yang sesuai dari Z^TZ.
6. Ambil nilai eigen λ_1, λ_2,..., λp dan urutkan dari yang terbesar ke terkecil. Dengan demikian, urutkan vektor eigen dalam P yang sesuai. (Misalnya, jika λ_2 adalah nilai eigen terbesar, ambil kolom kedua P dan letakkan di posisi kolom pertama.) Panggil matriks yang diurutkan dari vektor eigen P *. (Kolom P * harus sama dengan kolom P, tetapi mungkin dalam urutan yang berbeda.) Perhatikan bahwa vektor eigen ini tidak bergantung satu sama lain.
7. Hitung Z * = ZP *. Matriks baru ini, Z *, adalah versi X yang dipusatkan / terstandarisasi tetapi sekarang setiap pengamatan adalah kombinasi dari variabel asli, di mana bobot ditentukan oleh vektor eigen. Karena vektor eigen kita di P * tidak bergantung satu sama lain, setiap kolom Z * juga tidak bergantung satu sama lain!
8. Terakhir, kita perlu menentukan berapa banyak fitur yang harus dipertahankan dengan berapa banyak yang akan dihapus. Ada tiga metode umum untuk menentukan hal ini, yang dibahas di bawah ini dan diikuti dengan contoh eksplisit:

    a. Metode 1: Secara acak memilih berapa banyak dimensi yang ingin dipertahankan. Mungkin kita ingin merepresentasikan hal-hal secara visual dalam dua dimensi, jadi saya hanya dapat menyimpan dua fitur. Ini bergantung pada kasus penggunaan dan tidak ada aturan yang pasti tentang berapa banyak fitur yang harus saya pilih.
    b. Metode 2: Hitung proporsi varian yang dijelaskan untuk setiap fitur, pilih ambang batas, dan tambahkan fitur hingga Anda mencapai ambang tersebut.
    c. Metode 3: Ini terkait erat dengan Metode 2. Hitung proporsi varian yang dijelaskan untuk setiap fitur, urutkan fitur berdasarkan proporsi varian yang dijelaskan, dan plot proporsi kumulatif varian yang dijelaskan saat Anda menyimpan lebih banyak fitur.

Meskipun PCA adalah metode yang sangat teknis yang mengandalkan algoritme aljabar linier yang mendalam, PCA adalah metode yang relatif intuitif saat Anda memikirkannya.

1. Pertama, matriks kovariansi Z^TZ adalah matriks yang berisi perkiraan tentang bagaimana setiap variabel di Z berhubungan dengan setiap variabel lain di Z. Memahami bagaimana satu variabel dikaitkan dengan variabel lain cukup ampuh.
2. Kedua, nilai eigen dan vektor eigen penting. Vektor eigen mewakili arah. Pikirkan untuk memplot data Anda pada sebar multidimensi. Kemudian orang dapat menganggap vektor eigen individu sebagai "arah" tertentu dalam diagram sebar data Anda. Nilai eigen mewakili besaran, atau kepentingan. Nilai eigen yang lebih besar berkorelasi dengan arah yang lebih penting.

3. Ketiga, membuat asumsi bahwa lebih banyak variabilitas dalam arah tertentu berkorelasi dengan menjelaskan perilaku variabel dependen. Banyak variabilitas biasanya menunjukkan sinyal, sedangkan variabilitas kecil biasanya menunjukkan noise. Jadi, semakin banyak variabilitas yang ada dalam arah tertentu, secara teoritis, menunjukkan sesuatu yang penting yang ingin kita deteksi.

Jadi, PCA adalah metode yang menyatukan:

a. Ukuran bagaimana setiap variabel dikaitkan satu sama lain. (Matriks kovarian.)
b. Arah penyebaran data. (Vektor eigen.)
c. Kepentingan relatif dari arah yang berbeda ini. (Nilai Eigen.)

Import library

- Mengimport library yang akan digunakan
- Untuk **StandardScaler** dan **PCA** digunakan dari **sklearn**

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

Import dataset

- Menggunakan dataset dari https://archive.ics.uci.edu/ml/datasets/air+quality
- Menggunakan **pd.read_csv** karena dataset yang digunakan memiliki format **.csv**
- Karena dataset tidak memiliki nama kolom maka dibuatkan namanya dan dimasukkan menjadi parameter pada **pd.read_csv** sebagai **names**.

```
names = ['Date', 'Time', 'CO(GT)', 'PT08.S1(CO)', 'NMHC(GT)', 'C6H6(GT)', 'PT08.S2(NMHC)',
df = pd.read_csv("AirQualityUCI.csv")

print(df)
```

```
            Date      Time  CO(GT)  PT08.S1(CO)  ...  PT08.S5(O3)     T    RH      AH
0     3/10/2004  18:00:00     2.6         1360  ...         1268  13.6  48.9  0.7578
1     3/10/2004  19:00:00     2.0         1292  ...          972  13.3  47.7  0.7255
2     3/10/2004  20:00:00     2.2         1402  ...         1074  11.9  54.0  0.7502
3     3/10/2004  21:00:00     2.2         1376  ...         1203  11.0  60.0  0.7867
4     3/10/2004  22:00:00     1.6         1272  ...         1110  11.2  59.6  0.7888
...         ...       ...     ...          ...  ...          ...   ...   ...     ...
9352   4/4/2005  10:00:00     3.1         1314  ...         1729  21.9  29.3  0.7568
9353   4/4/2005  11:00:00     2.4         1163  ...         1269  24.3  23.7  0.7119
9354   4/4/2005  12:00:00     2.4         1142  ...         1092  26.9  18.3  0.6406
9355   4/4/2005  13:00:00     2.1         1003  ...          770  28.3  13.5  0.5139
```

```
      9356   4/4/2005  14:00:00     2.2         1071  ...          816  28.5  13.1  0.5028

      [9357 rows x 15 columns]
```

Standarisasi data

- Melakukan standarisasi karena PCA dipengaruhi oleh skala
- Menggunakan **StandardScaler** untuk standarisasi
- Standarisasi fitur set data ke dalam skala unit (mean=0 dan variance=1)

```
# mengambil fitur-fitur pada dataset
features = names[2:]

# memisahkan fitur dan label
x = df.loc[:, features].values
y = df.loc[:, ['PT08.S1(CO)']].values

# Standarisasi fitur
x = StandardScaler().fit_transform(x)
```

Proyeksi PCA ke 2D

- Mereduksi menjadi 2 fitur/2 dimensi dengan PCA

```
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
principalData = pd.DataFrame(data = principalComponents, columns=['principal component 1',


finalData = pd.concat([principalData, df[['PT08.S1(CO)']]], axis=1)
finalData
```

	principal component 1	principal component 2	PT08.S1(CO)
0	-1.365151	0.569067	1360
1	-0.907059	-0.064023	1292
2	-1.089141	0.193355	1402
9352	-1.120703	2.195673	1314

Visualisasi proyeksi 2D

- Memvisualisais dengan scatter plot untuk tiap class

```
fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel("Principal Component 1", fontsize=15)
ax.set_ylabel("Principal Component 2", fontsize=15)
ax.set_title("2 Component PCA", fontsize=20)
targets = list(df['PT08.S1(CO)'].unique())
colors = ["red","green","blue","yellow","pink","black","orange","purple","beige","brown","
          "#43d023","#352601","#3c4e4b","#99dac4","#546217","#99bbcf","#f3014e","#98c5b4",
          "#3c4e4b","#7fdf78","#ea6d70","#a44c79","#3acbc3"]
for target, color in zip(targets, colors):
  indicesToKeep = finalData['PT08.S1(CO)'] == target
  ax.scatter(finalData.loc[indicesToKeep, 'principal component 1'], finalData.loc[indicesT
ax.legend(targets)
ax.grid()
```

2 Component PCA

Varians yang Dijelaskan

```
- Komponen utama 1 berisi 50,65% varian
- Komponen utama 2 berisi 22,65% varian
- Total informasi dari kedua komponen tersebu
- Hasil pada PCA cukup baik karena banyak inf
```

- Komponen utama 1 berisi 50,65% varian
- Komponen utama 2 berisi 22,65% varian
- Total informasi dari kedua komponen tersebut adalah 73,30%
- Hasil pada PCA cukup baik karena banyak informasi sesuai dari data aslinya

```
pca.explained_variance_ratio_
```

```
array([0.50652352, 0.22652556])
```

```
pca.get_covariance()
```

```
array([[ 0.6599734 ,  0.12560183,  0.07187144, -0.08574106,  0.17803661,
         0.50133708, -0.36575196,  0.45902691,  0.02546129,  0.25459881,
        -0.15672337, -0.13840915, -0.15006707],
       [ 0.12560183,  1.19804139,  0.11585985,  0.83821687,  0.84356192,
         0.2819798 ,  0.18439974,  0.21855335,  0.78299947,  0.78384513,
         0.76561048,  0.73969998,  0.77233238],
       [ 0.07187144,  0.11585985,  0.3401235 ,  0.07527252,  0.12057766,
         0.11523208, -0.04226765,  0.10126097,  0.08821424,  0.12716286,
         0.05545548,  0.05578503,  0.05730452],
       [-0.08574106,  0.83821687,  0.07527252,  1.24038948,  0.76481256,
        -0.02173875,  0.42125335, -0.06108462,  0.79767166,  0.65468882,
         0.89398455,  0.85556116,  0.89679057],
       [ 0.17803661,  0.84356192,  0.12057766,  0.76481256,  1.13213808,
         0.35141298,  0.10666572,  0.28487745,  0.73317043,  0.77463778,
         0.68464392,  0.66378336,  0.69208098],
       [ 0.50133708,  0.2819798 ,  0.11523208, -0.02173875,  0.35141298,
         1.05695229, -0.49467498,  0.67416134,  0.12876145,  0.45245958,
        -0.13058859, -0.10779547, -0.12039158],
       [-0.36575196,  0.18439974, -0.04226765,  0.42125335,  0.10666572,
        -0.49467498,  0.82457153, -0.46797725,  0.26680458, -0.00791256,
         0.47888047,  0.44705731,  0.47344363],
       [ 0.45902691,  0.21855335,  0.10126097, -0.06108462,  0.28487745,
         0.67416134, -0.46797725,  0.93036325,  0.08124987,  0.38154682,
        -0.1585352 , -0.13612224, -0.14940218],
       [ 0.02546129,  0.78299947,  0.08821424,  0.79767166,  0.73317043,
         0.12876145,  0.26680458,  0.08124987,  1.03285747,  0.65779407,
         0.74921979,  0.72043885,  0.75368316],
       [ 0.25459881,  0.78384513,  0.12716286,  0.65468882,  0.77463778,
         0.45245958, -0.00791256,  0.38154682,  0.65779407,  1.07437598,
         0.56363636,  0.55025975,  0.57210313],
       [-0.15672337,  0.76561048,  0.05545548,  0.89398455,  0.68464392,
        -0.13058859,  0.47888047, -0.1585352 ,  0.74921979,  0.56363636,
         1.19585858,  0.83995973,  0.88153192],
       [-0.13840915,  0.73969998,  0.05578503,  0.85556116,  0.66378336,
```

```
               -0.10779547,  0.44705731, -0.13612224,  0.72043885,  0.55025975,
                0.83995973,  1.11734625,  0.84134088],
              [-0.15006707,  0.77233238,  0.05730452,  0.89679057,  0.69208098,
               -0.12039158,  0.47344363, -0.14940218,  0.75368316,  0.57210313,
                0.88153192,  0.84134088,  1.19839828]])
```

```
pca.components_
```

```
array([[ 0.00410216, -0.36375425, -0.03827969, -0.38057229, -0.33777714,
        -0.03732021, -0.14306991, -0.01687156, -0.33744342, -0.29861058,
        -0.36102173, -0.34657784, -0.36281723],
       [ 0.36186868,  0.14182505,  0.07656327, -0.07981737,  0.19622582,
         0.52785618, -0.38049605,  0.48284036,  0.03587774,  0.27562522,
        -0.15494002, -0.1360843 , -0.14789649]])
```

```
pca.explained_variance_ratio_.sum()
```

```
0.7330490761372497
```

✓ 0s completed at 11:13 PM