

✓ LAPORAN TUGAS PRAKTIKUM FEATURE EXTRACTION METODE PCA

Laporan diperuntukan sebagai pemenuh syarat penilaian praktikum mata kuliah Data Mining

Dosen Pengampu : Illham Faishal Mahdy, S.Stat., M.Stat.

Oleh : Catherine V. Pang 2C2220008 VI/A

Hari, Tanggal: Senin, 17 April 2025

Pertemuan Ke: 2

PROGRAM STUDI S1 SAINS DATA FAKULTAS SAINS DAN TEKNOLOGI UNIVERSITAS KOPERASI INDONESIA 2025

1. Mengimpor pustaka yang diperlukan

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

Memuat dataset

```
data = pd.read_csv("/content/2021socio_economic_indonesia.csv")
data
```



	province	cities_reg	poorpeople_percentage	reg_gdp	life_exp	avg_schooltime	exp_percap
0	Aceh	Simeulue	18.98	2.275	65.240	9.48	7148
1	Aceh	Aceh Singkil	20.36	2.425	67.355	8.68	8776
2	Aceh	Aceh Selatan	13.18	5.531	64.360	8.88	8180
3	Aceh	Aceh Tenggara	13.41	5.063	68.155	9.67	8030
4	Aceh	Aceh Timur	14.45	10.616	68.705	8.21	8577
...
509	Papua	Puncak	36.26	1.438	66.060	2.16	5412
510	Papua	Dogiyai	28.81	1.356	65.845	4.94	5415
511	Papua	Intan Jaya	41.66	1.274	65.580	3.09	5328
512	Papua	Deiyai	40.59	1.439	65.355	3.25	4673
513	Papua	Kota Jayapura	11.39	32.207	70.455	11.57	14937

514 rows × 7 columns

Identifikasi missing values

```
print("\nMissing Values:")
missing_values = (data == '?').sum()
print(missing_values)
```



```
Missing Values:
province          0
cities_reg        0
poorpeople_percentage  0
reg_gdp           0
life_exp          0
avg_schooltime    0
exp_percap        0
dtype: int64
```

```
processed_data = data.copy()
```

```
processed_data = processed_data.replace('?', np.nan)
```

Numerical features - Mean Imputation

Update with the actual numerical columns in your dataset

```
numerical_features = processed_data.select_dtypes(include=np.number).columns
```

Remove 'normalized-losses', 'bore', 'stroke', 'horsepower', 'peak-rpm', 'price'

and replace them with your actual numerical columns

Impute missing values for numerical features


for feature in numerical_features:

```
    processed_data[feature] = pd.to_numeric(processed_data[feature], errors='coerce')
```

```
    processed_data[feature].fillna(processed_data[feature].mean(), inplace=True)
```

Categorical features - Mode Imputation

```
# Update with the actual categorical columns in your dataset, if any
categorical_features = processed_data.select_dtypes(exclude=np.number).columns
# Remove 'num-of-doors' and replace it with your actual categorical columns, if any
# Impute missing values for categorical features
for feature in categorical_features:
    processed_data[feature].fillna(processed_data[feature].mode()[0], inplace=True)
```

 <ipython-input-4-5303ef7bc72c>:10: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col]

```
processed_data[feature].fillna(processed_data[feature].mean(), inplace=True)
<ipython-input-4-5303ef7bc72c>:18: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting
```

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col]

```
processed_data[feature].fillna(processed_data[feature].mode()[0], inplace=True)
```

```
processed_data.head()
```




	province	cities_reg	poorpeople_percentage	reg_gdp	life_exp	avg_schooltime	exp_percap
0	Aceh	Simeulue	18.98	2.275	65.240	9.48	7148
1	Aceh	Aceh Singkil	20.36	2.425	67.355	8.68	8776
2	Aceh	Aceh Selatan	13.18	5.531	64.360	8.88	8180
3	Aceh	Aceh Tenggara	13.41	5.063	68.155	9.67	8030
4	Aceh	Aceh Timur	14.45	10.616	68.705	8.21	8577

```
# Separate numerical and categorical features (after one-hot encoding)
numerical_features = processed_data.select_dtypes(include=np.number).columns
categorical_features = processed_data.select_dtypes(exclude=np.number).columns
```

```
# Create a new DataFrame with only the numerical features
numerical_data = processed_data[numerical_features]
```

```
# Now 'numerical_data' contains only the numerical features, excluding the categorical ones.
```

```
numerical_data.head()
```



	poorpeople_percentage	reg_gdp	life_exp	avg_schooltime	exp_percap
0	18.98	2.275	65.240	9.48	7148
1	20.36	2.425	67.355	8.68	8776
2	13.18	5.531	64.360	8.88	8180
3	13.41	5.063	68.155	9.67	8030
4	14.45	10.616	68.705	8.21	8577

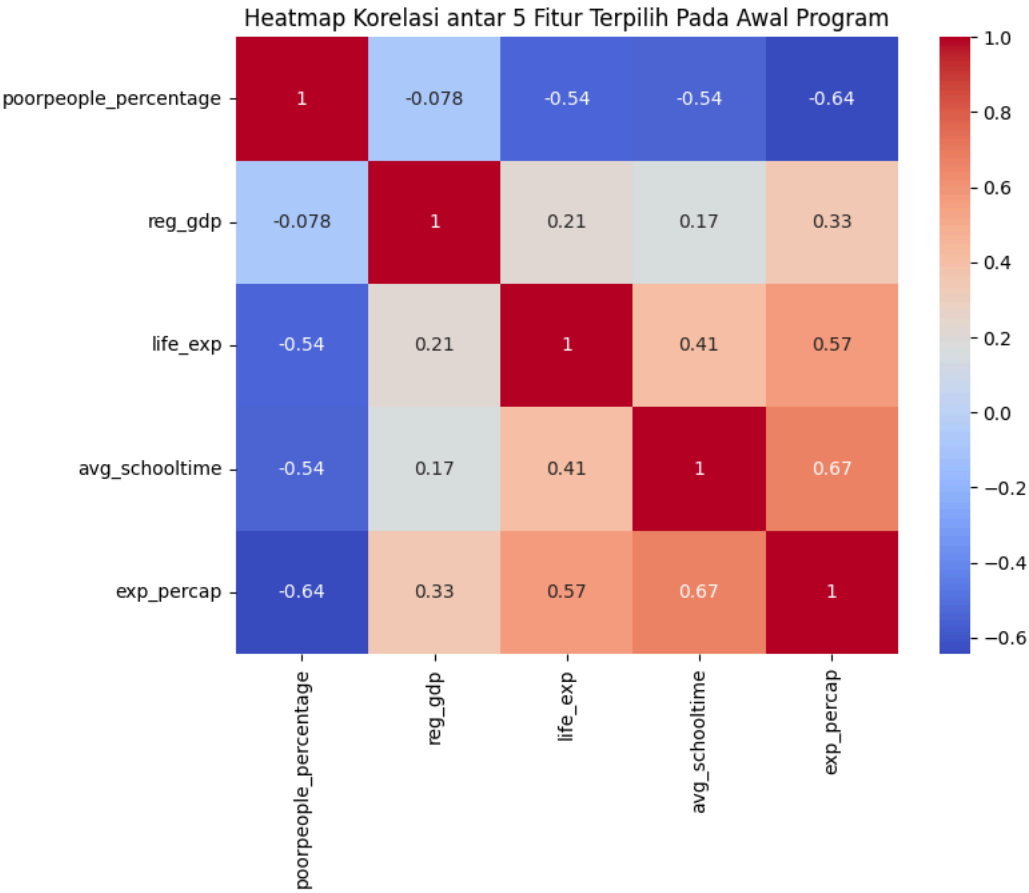
```
#Korelasi Antarfitur
```

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Pilih hanya 5 fitur numerik yang digunakan dalam PCA
selected_features = ['poorpeople_percentage', 'reg_gdp', 'life_exp', 'avg_schooltime', 'exp_percap']
filtered_data = processed_data[selected_features]
```

```
# Hitung korelasi
corr_matrix = filtered_data.corr()
```

```
# Visualisasikan dengan heatmap
plt.figure(figsize=(8,6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title("Heatmap Korelasi antar 5 Fitur Terpilih Pada Awal Program")
plt.show()
```



Interpretasi Heatmap:

Pasangan Fitur	Koefisien Korelasi	Interpretasi
poorpeople_percentage vs reg_gdp	-0.08	Hampir tidak ada hubungan linear—pertumbuhan GDP per kapita tidak serta-merta menurunkan kemiskinan.
poorpeople_percentage vs life_exp	-0.54	Korelasi negatif sedang—daerah dengan persentase kemiskinan tinggi cenderung memiliki harapan hidup rendah.
poorpeople_percentage vs avg_schooltime	-0.54	Korelasi negatif sedang—kemiskinan tinggi berkaitan dengan lama sekolah rata-rata yang lebih pendek.
poorpeople_percentage vs exp_percap	-0.64	Korelasi negatif kuat—semakin tinggi belanja per kapita, semakin rendah persentase kemiskinan.
reg_gdp vs life_exp	0.21	Korelasi positif lemah—GDP yang lebih tinggi ada kaitannya, meski tidak kuat, dengan harapan hidup.
reg_gdp vs avg_schooltime	0.17	Korelasi positif lemah—GDP tinggi sedikit berkaitan dengan lama sekolah.
reg_gdp vs exp_percap	0.33	Korelasi positif sedang—GDP lebih besar umumnya berarti belanja per kapita lebih besar.
life_exp vs avg_schooltime	0.41	Korelasi positif sedang—lama sekolah rata-rata lebih panjang cenderung berhubungan dengan harapan hidup lebih panjang.
life_exp vs exp_percap	0.57	Korelasi positif sedang—kuat—daerah dengan belanja per kapita tinggi biasanya punya harapan hidup lebih baik.
avg_schooltime vs exp_percap	0.67	Korelasi positif kuat—lama sekolah dan daya beli (belanja per kapita) berjalan seiring.

Insight Utama

1. `poorpeople_percentage` **sangat anticorrelated** dengan indikator kesejahteraan lain (terutama `exp_percap`): ini menunjukkan bahwa fitur kemiskinan memang membawa “arah” informasi yang berlawanan dengan GDP, belanja per kapita, harapan hidup, dan pendidikan.
2. **Keempat fitur lain** (`reg_gdp`, `life_exp`, `avg_schooltime`, `exp_percap`) saling berkorelasi positif—terutama antara `avg_schooltime` & `exp_percap` (0.67) serta `life_exp` & `exp_percap` (0.57)—membentuk cluster “indikator kesejahteraan” yang sejalan.
3. Karena `poorpeople_percentage` anticorrelated dan relatif redundant (menggambarkan aspek yang berlawanan), fitur ini lebih baik **dihapus** sebelum PCA untuk memudahkan interpretasi komponen utama yang fokus pada sisi “positif” pembangunan.

```
# 3. Standarisasi data
X = numerical_data.iloc[:, 1:]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print(X_scaled)
```



```
[[ -0.38684361 -1.26836081  0.64040648 -1.17030327]
 [ -0.38505946 -0.65576966  0.14938439 -0.57056109]
 [ -0.34811563 -1.52324507  0.27213991 -0.79012272]
 ...
 [ -0.39874984 -1.1698828  -3.28163241 -1.84077671]
 [ -0.39678728 -1.23505207 -3.18342799 -2.08207347]
 [ -0.03082219  0.24211807  1.92320167  1.69910203]]
```

```
# Menghitung Matriks Kovarians
cov_matrix = np.cov(X_scaled, rowvar=False)
print("Matriks Kovarians:")
print(cov_matrix)
```

```
↳ Matriks Kovarians:
[[1.00194932 0.20955483 0.16572578 0.33460292]
 [0.20955483 1.00194932 0.41531853 0.56687808]
 [0.16572578 0.41531853 1.00194932 0.67015105]
 [0.33460292 0.56687808 0.67015105 1.00194932]]
```

```
# Membuat DataFrame dari matriks kovarians
cov_df = pd.DataFrame(cov_matrix, columns=X.columns, index=X.columns)
```

```
# Menampilkan DataFrame
print("\nDataFrame Matriks Kovarians:")
cov_df
```

```
↳ DataFrame Matriks Kovarians:
```

	reg_gdp	life_exp	avg_schooltime	exp_percap
reg_gdp	1.001949	0.209555	0.165726	0.334603
life_exp	0.209555	1.001949	0.415319	0.566878
avg_schooltime	0.165726	0.415319	1.001949	0.670151
exp_percap	0.334603	0.566878	0.670151	1.001949

```
# Menghitung Eigenvalues dan Eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)

# Membuat DataFrame untuk Eigenvalues
eigenvalue_df = pd.DataFrame({'Eigenvalue': eigenvalues})

# Menampilkan DataFrame Eigenvalues
print("\nEigenvalues:")
eigenvalue_df
```

```
↳ Eigenvalues:
```

	Eigenvalue
0	2.248561
1	0.882401
2	0.281745
3	0.595092

```
# Membuat DataFrame untuk Eigenvectors
eigenvector_df = pd.DataFrame(eigenvectors, index=X.columns)

# Menampilkan DataFrame Eigenvectors
print("\nEigenvectors:")
eigenvector_df
```

```
↳ Eigenvectors:
```

	0	1	2	3
reg_gdp	0.316925	0.931420	-0.164430	0.070558
life_exp	0.504166	-0.155131	-0.254781	-0.810455
avg_schooltime	0.533150	-0.320075	-0.543539	0.563798
exp_percap	0.600938	-0.077097	0.782697	0.142532

```
var_exp = [(i / sum(eigenvalues)) for i in sorted(eigenvalues, reverse=True)]
cum_var_exp = np.cumsum(var_exp)
print("Varians Kumulatif")
print(cum_var_exp)
```

```
↳ Varians Kumulatif
[0.56104652 0.78121747 0.92970091 1.        ]
```

```
n_components = np.argmax(cum_var_exp >= 0.85) + 1
print(f"Jumlah komponen utama yang dipilih: {n_components}")
```

Jumlah komponen utama yang dipilih: 3

```
# 4. Melakukan PCA
pca = PCA(n_components=n_components)
X_pca = pca.fit_transform(X_scaled)
```

```
# Mengambil eigenvalues dan eigenvectors
eigenvalues = pca.explained_variance_
# Mengambil eigenvalues dan eigenvectors
eigenvalues = pca.explained_variance_
eigenvectors = pca.components_
```

```
print(eigenvectors)
```

```
[[ 0.31692524  0.50416577  0.5331502   0.60093771]
 [ 0.93141963 -0.15513114 -0.32007469 -0.07709733]
 [-0.07055846  0.81045524 -0.5637982  -0.1425321 ]]
```

```
eigenvector_df = pd.DataFrame(eigenvectors, columns=X.columns)
eigenvector_df
```

	reg_gdp	life_exp	avg_schooltime	exp_percap
0	0.316925	0.504166	0.533150	0.600938
1	0.931420	-0.155131	-0.320075	-0.077097
2	-0.070558	0.810455	-0.563798	-0.142532

```
# 5. Skor Komponen Utama
df_scores = pd.DataFrame(X_pca, columns=[f"PC{i}" for i in range(1, n_components + 1)])
print(df_scores)
```

	PC1	PC2	PC3
0	-1.123911	-0.278302	-1.194909
1	-0.715879	-0.260747	-0.507202
2	-1.208018	-0.114128	-1.250774
3	-0.430300	-0.440770	-0.625039
4	-0.685721	-0.132675	-0.024085
...
509	-3.787006	1.162935	1.622497
510	-2.908331	0.625458	0.609929
511	-3.571984	1.002368	1.192546
512	-3.696865	1.001477	1.118616
513	2.158709	-0.812833	-1.128074

[514 rows x 3 columns]

✓ Rangkuman hasil perhitungan PCA, beserta komponen utama (PC) terpilih, persamaan linearnya, dan proporsi varians yang dijelaskan:

1. Jumlah Komponen Terpilih

Pada program awal dipilih 3 komponen utama, karena kumulatif variansnya baru melewati 85 % pada PC3:

```
n_components = np.argmax(cum_var_exp >= 0.85) + 1 # = 3
```

2. Persamaan Komponen Utama

Misalkan variabel asli adalah:

- reg_gdp
- life_exp
- avg_schooltime
- exp_percap

Dengan hasil perhitungan Eigen Vector-nya adalah:

Komponen	reg_gdp	life_exp	avg_schooltime	exp_percap
PC1	0.316925	0.504166	0.533150	0.600938
PC2	0.931420	-0.155131	-0.320075	-0.077097

Komponen	reg_gdp	life_exp	avg_schooltime	exp_percap
PC3	-0.070558	0.810455	-0.563798	-0.142532

Maka tiga komponen terpilih (PC1-PC3) adalah kombinasi linier berikut (loading = koefisien pada eigenvector):

Komponen	Persamaan
PC1	$0.3169 \cdot \text{reg_gdp} + 0.5042 \cdot \text{life_exp} + 0.5332 \cdot \text{avg_schooltime} + 0.6009 \cdot \text{exp_percap}$
PC2	$0.9314 \cdot \text{reg_gdp} - 0.1551 \cdot \text{life_exp} - 0.3201 \cdot \text{avg_schooltime} - 0.0771 \cdot \text{exp_percap}$
PC3	$-0.0706 \cdot \text{reg_gdp} + 0.8105 \cdot \text{life_exp} - 0.5638 \cdot \text{avg_schooltime} - 0.1425 \cdot \text{exp_percap}$

Catatan: Semua variabel sudah distandarisasi (mean = 0, std = 1) sebelum dihitung PCA.

3. Proporsi Varians per Komponen

Eigenvalues (dari covariance matrix) dibagi total eigenvalues (≈ 4.0078) memberi proporsi varians:

Komponen	Eigenvalue	Proporsi Varians	Varians Kumulatif
PC1	2.2486	$2.2486/4.0078 \approx 56.1 \%$	56.1 %
PC2	0.8824	$0.8824/4.0078 \approx 22.0 \%$	78.1 %
PC3	0.5951	$0.5951/4.0078 \approx 14.8 \%$	92.9 %
PC4	0.2817	$0.2817/4.0078 \approx 7.0 \%$	100 %

Karena hingga PC2 kumulatif baru $\sim 78.1 \%$, kamu butuh **PC3** agar mencapai $\geq 85 \%$ (tepatnya $\sim 92.9 \%$).

4. Interpretasi singkat

- **PC1:** muatan positif pada ke-4 variabel, jadi ini menggambarkan "dimensi umum kesejahteraan/regional output" (GDP, harapan hidup, sekolah, pengeluaran per kapita).
- **PC2:** didominasi `reg_gdp` (0.93), artinya membedakan provinsi/kabupaten dengan GDP tinggi tapi indikator sosial lainnya rendah.
- **PC3:** membedakan daerah dengan harapan hidup tinggi & rata-rata sekolah panjang, tetapi belanja per kapita/GDP relatif rendah.

Dengan demikian, cukup mengambil **PC1–PC3** sebagai fitur baru—mereka sudah merangkum $\sim 93 \%$ informasi varians dari 4 variabel asli.

✓ Berikut adalah simulasi perhitungan manual PCA dari dataset yang sama dengan praktikum kali ini.

https://github.com/riinndescartes/Data-Mining/blob/main/Simulasi_PCA_2x2.ipynb