

Álgebra Linear - PCA

Fernanda Rafaela

Novembro, 2024

1 Dataset

O dataset inclui fatores de risco que auxiliam na previsão e prevenção de doenças cardiovasculares, possibilitando a análise de elementos que influenciam a mortalidade por insuficiência cardíaca.

Link: <https://www.kaggle.com/datasets/andrewmvd/heart-failure-clinical-data>

2 Desenvolvimento

Para realizar a normalização dos dados e aplicar o algoritmo de PCA, serão usadas as seguintes bibliotecas no Python:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Figure 1: Bibliotecas utilizadas para análise

Primeiro, importamos o arquivo CSV e normalizamos o dataset, calculando a média de cada coluna e subtraindo do total:

```
6 df = pd.read_csv('heart_failure_clinical_records_dataset.csv')
7 df_scaled = (df - df.mean()) / df.std()
8
```


	age	anaemia	creatinine_phosphokinase	...	smoking	time	DEATH_EVENT
0	1.190949	-0.869647	0.000165	...	-0.686531	-1.626775	1.451727
1	-0.490457	-0.869647	7.502063	...	-0.686531	-1.601007	1.451727
2	0.350246	-0.869647	-0.449186	...	1.451727	-1.588122	1.451727
3	-0.910808	1.146046	-0.485257	...	-0.686531	-1.588122	1.451727
4	0.350246	1.146046	-0.434757	...	-0.686531	-1.575238	1.451727

[5 rows x 13 columns]

Figure 2: Importação do CSV

Em seguida, calculamos a matriz de covariância, transpondo-a:

```
9 cov = np.cov(df_scaled.T)
```

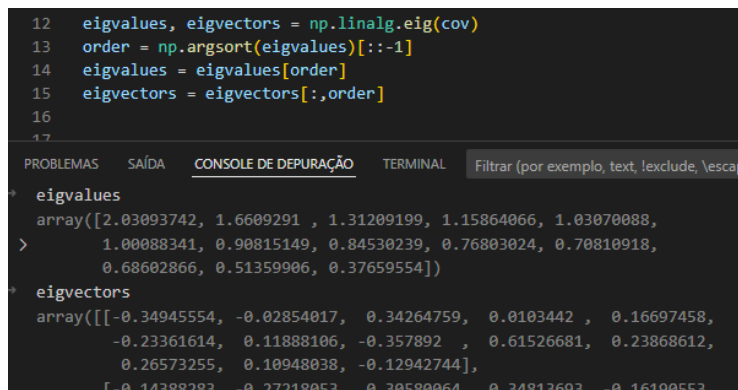


```
array([[ 1.          ,  0.08800644, -0.0815839 , -0.10101239,  0.06009836,
         0.09328868, -0.05235437,  0.15918713, -0.04596584,  0.06542952,
         0.01866787, -0.22406842,  0.25372854],
       [ 0.08800644,  1.          , -0.19074103, -0.01272905,  0.03155697,
         0.038182 , -0.04378555,  0.0521736 ,  0.04188161, -0.09476896,
        -0.10728984, -0.14141398,  0.0662701 ],
       [-0.0815839 , -0.19074103,  1.          , -0.00963851, -0.04407955,
        -0.07058998,  0.02446339, -0.01640848,  0.05955016,  0.07979063,
         0.00242124, -0.00934565,  0.06272816],
       [-0.10101239, -0.01272905, -0.00963851,  1.          , -0.00485031,
```

Figure 3: Matriz de Covariância

Agora, calculamos os autovetores e autovalores e ordenamos do maior para o menor:

```
12 eigvalues, eigvectors = np.linalg.eig(cov)
13 order = np.argsort(eigvalues)[::-1]
14 eigvalues = eigvalues[order]
15 eigvectors = eigvectors[:,order]
```



```
array([2.03093742, 1.6609291 , 1.31209199, 1.15864066, 1.03070088,
       1.00088341, 0.90815149, 0.84530239, 0.76803024, 0.70810918,
       0.68602866, 0.51359906, 0.37659554])
array([[ -0.34945554, -0.02854017,  0.34264759,  0.0103442 ,  0.16697458,
        -0.23361614,  0.11888106, -0.357892 ,  0.61526681,  0.23868612,
         0.26573255,  0.10948038, -0.12942744],
       [ -0.14388283, -0.27218053,  0.30580064,  0.34813693, -0.16190553,
         0.22555342, -0.56627826,  0.06160122, -0.21708909,  0.02884298,
         0.4778976 ,  0.09401432,  0.0613532 ]])
array([2.03093742, 1.6609291 ])
```

Figure 4: Cálculo dos Autovalores e Autovetores

Maiores autovalores e autovetores:

```
> eigvectors[:2]
array([[ -0.34945554, -0.02854017,  0.34264759,  0.0103442 ,  0.16697458,
        -0.23361614,  0.11888106, -0.357892 ,  0.61526681,  0.23868612,
         0.26573255,  0.10948038, -0.12942744],
       [ -0.14388283, -0.27218053,  0.30580064,  0.34813693, -0.16190553,
         0.22555342, -0.56627826,  0.06160122, -0.21708909,  0.02884298,
         0.4778976 ,  0.09401432,  0.0613532 ]])
> eigvalues[0:2]
array([2.03093742, 1.6609291 ])
```

Figure 5: Exibição dos maiores autovetores e autovalores

Usando os 2 maiores autovalores calculamos o pca e fazemos um grafico bidimensional:

```
21 k = 2
22 pca = np.matmul(df_scaled, eigvectors[:, :k])
23
24 plt.figure(figsize=(8, 6))
25 sns.scatterplot(x=pca.iloc[:, 0], y=pca.iloc[:, 1])
26 plt.title('PCA: PC1 vs PC2')
27 plt.xlabel('Componente Principal 1')
28 plt.ylabel('Componente Principal 2')
29 plt.show()
```

Figure 6: Criação mapa 2d

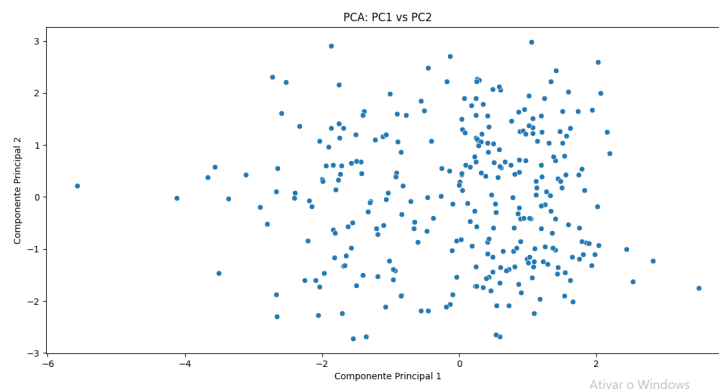


Figure 7: Mapa 2d

Usando os maiores 3 autovetores calculamos o pca e fazemos o gráfico tridimensional:

```
k = 3
pca = np.matmul(df_scaled, eigvectors[:, :k])

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(pca.iloc[:, 0], pca.iloc[:, 1], pca.iloc[:, 2])
ax.set_xlabel('Componente Principal 1')
ax.set_ylabel('Componente Principal 2')
ax.set_zlabel('Componente Principal 3')
plt.title('PCA: PC1 vs PC2 vs PC3')
plt.show()
```

Figure 8: Criação mapa 3d

PCA: PC1 vs PC2 vs PC3

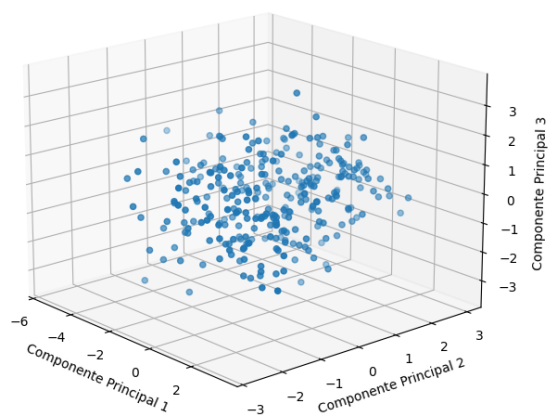


Figure 9: Mapa 3d

repo: <https://github.com/xfehrxx/pca>