Análise PCA

Jessica Barreto
Lethicia Costa

Junho de 2023

1 Introdução

Para a seguinte pesquisa, temos alguns objetivos que devem ser seguidos: análise de PCA dos dados, explicar e mostrar a dimensão dos dados, mostrar matriz de covariância, autovalores e autovetores de cada coluna, mostrar os maiores autovalores e seus autovetores, plot e discussões.

Covariância é uma medida estatistica que possibilita comparar duas variaveis, permitindo entender como elas se relacionam entre si(SUNO)

A análise de componetes principais ou PCA (Principal Component Analysis) é uma tecnica multivariada que pode ser usada para analisar inter-relações entre grandes números de variaveis e explicar essa variaveis em termos de seus componentes(STATPLACE)

2 Banco de Dados

O dataset escolhido contém uma amostra razoavelmente pequena de modo a ser utilizada para uma analise superficial, é entendido que para uma analise apronfundade seria necessário uma base com mais dados, porém podemos passar por cada objetivo já citado.

3 Código Python

article verbatim pandas numpy seaborn matplotlib.pyplot

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

label_bream = 0
label_smelt = 1

dados = pd.read_csv("/content/Fish.csv", sep=",")

display(dados)
```

Antes de qualquer analise abrimos o arquivo csv, após usamos o codigo a baixo para saber a correlação dos dados.

```
#calculando a matriz de correlação

corr = dados.iloc[:,0:9].corr()
corr
```

```
\usepackage{seaborn}
\usepackage{matplotlib.pyplot}
```

```
\usepackage{pandas}

# Selecionar apenas as colunas numéricas relevantes
numeric_cols = ['Weight', 'Height', 'Width']
numeric_dados = dados[numeric_cols]

# Calcular a matriz de correlação para as colunas numéricas
corr_matrix = numeric_dados.corr()

# Plotar o mapa de calor
\texttt{plt.figure(figsize=(10, 8))}
heatmap = \texttt{sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')}

# Configurar os rótulos dos eixos
heatmap.set\_xticklabels(heatmap.get\_xticklabels(), rotation=45)
heatmap.set\_yticklabels(heatmap.get\_yticklabels(), rotation=0)

# Exibir o mapa de calor
\texttt{plt.show()}
```

Utilizamos o codigo anterior para exibir um mapa de calor com as colunas weight, height e width.

article graphicx

4 Mapa de Calor

Foi escolhido o mapa de calor para visualizar os dados e se apronfundar no comportamento dos dados

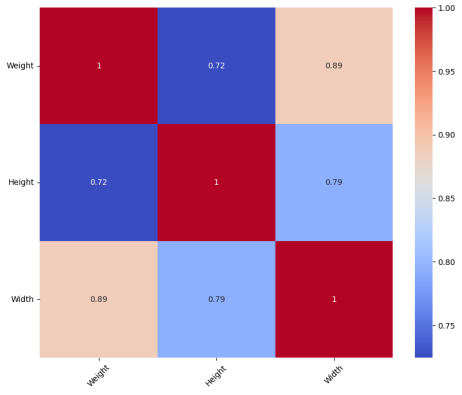

Figure 1: Mapa de Calor.

Referência à Figura **??**.

```
\usepackage{numpy}

\begin{verbatim}
import numpy as np

# Selecionar apenas as colunas numéricas relevantes
numeric_cols = ['Weight', 'Height', 'Width']
dados_numeric = dados[numeric_cols]

# Converter as colunas numéricas para valores float
dados_numeric = dados_numeric.astype(float)

# Calcular a matriz de covariância
cov_matrix = np.cov(dados_numeric, rowvar=False)

# Exibir a matriz de covariância
\texttt{print("Matriz de Covariância:")}
\texttt{print(cov_matrix)}
```

O resultado do código foi a seguinte matriz de covariância
Matriz de Covariância: 1.28148475e+05 1.11141330e+03 5.34990098e+02
1.11141330e+03 1.83715758e+01 5.72912464e+00
5.34990098e+02 5.72912464e+00 2.84193469e+00
(0)

article amsmath graphicx

5 PCA

```
from sklearn.decomposition import PCA
import pandas as pd

# Selecionar apenas as colunas numéricas relevantes
numeric_cols = ['Weight', 'Height', 'Width']
numeric_dados = dados[numeric_cols]

# Inicializar o objeto PCA
pca = PCA(n_components=2)

# Aplicar a PCA aos dados
pca_data = pca.fit_transform(numeric_dados)

# Imprimir os resultados
print(pca_data)
```

Os resultados obtidos foram os seguintes:

[[-1.56298725e+02 -3.90461808e+00] [-1.08291430e+02 -4.45473931e+00] [-5.82930024e+01 -3.94721140e+00] [-3.52920177e+01 -4.05464908e+00] [3.17052305e+01 -3.24706318e+00] [5.17134884e+01 -4.18563535e+00] [1.01717643e+02 -4.34641684e+00] [-8.29280993e+00 -3.77889916e+00] [5.17166303e+01 -4.57375477e+00] [1.01716719e+02 -4.35144152e+00] [7.67187960e+01 -4.63483949e+00] [1.01717371e+02 -4.47608663e+00] [1.01710197e+02 -3.81083499e+00] [-5.82781158e+01 -5.51829334e+00] [2.01719282e+02 -4.18618212e+00] [2.01725184e+02 -4.71905647e+00] [3.01714312e+02 -3.19355373e+00] [3.01714618e+02 -3.25900745e+00] [2.11724549e+02 -4.76267071e+00] [6.01683851e+02 1.87261104e+00] [-3.43337657e+02 -7.61110893e-01] [-3.38340284e+02 -4.45810546e-01] [-3.08332994e+02 -1.03921900e+00] [-2.78324801e+02 -1.77541159e+00] [-2.48320284e+02 -2.04631370e+00] [-2.58322895e+02 -1.78581944e+00] [-3.90882611e+02 3.79042199e+00] [-3.91384749e+02 4.02886855e+00] [-3.88679852e+02 3.55973747e+00] [-3.88581264e+02 3.68590545e+00] [-3.89682091e+02 3.77889879e+00] [-3.88380112e+02 3.55729709e+00] [-3.88480600e+02 3.57177878e+00] [-3.88580751e+02 3.58256789e+00] [-3.86180827e+02 3.68551285e+00] [-3.84978457e+02 3.37607903e+00] [-3.86179784e+02 3.51851334e+00] [-3.78671577e+02 2.89046295e+00] [-3.78471855e+02 2.85795309e+00]]

```
   com o pca.component_:
 array([[ 0.99995367,  0.00867308,  0.00417467],
       [ 0.00914402, -0.99139481, -0.13058607]])
```

6 Autovalores e Autovetores

```
 # Calcular a matriz de covariância
cov_matrix = np.cov(numeric_dados, rowvar=False)

# Calcular os autovalores e autovetores
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)

# Imprimir os autovalores e autovetores
for i in range(len(eigenvalues)):
    print("Autovalor", i+1, ":", eigenvalues[i])
    print("Autovetor", i+1, ":", eigenvectors[:, i])
```

```
   Resultado: Autovalor 1 : 128160.34845397402
Autovetor 1 : [0.99995367 0.00867308 0.00417467]
Autovalor 2 : 8.875211259928788
Autovetor 2 : [ 0.00914402 -0.99139481 -0.13058607]
Autovalor 3 : 0.46496582705991896
Autovetor 3 : [-0.00300616 -0.1306182   0.99142819]

Acrescentamos um código para uma ordenação descrescente
# Ordenar os autovalores em ordem decrescente
sorted_indices = np.argsort(eigenvalues)[::-1]
sorted_eigenvalues = eigenvalues[sorted_indices]
```

```
sorted_eigenvectors = eigenvectors[:, sorted_indices]

# Imprimir os maiores autovalores e seus autovetores
num_components = 2  # Número de componentes principais a serem considerados
for i in range(num_components):
    print("Autovalor", i+1, ":", sorted_eigenvalues[i])
    print("Autovetor", i+1, ":", sorted_eigenvectors[:, i])

    Resultado: Autovalor 1 : 128160.34845397402
Autovetor 1 : [0.99995367 0.00867308 0.00417467]
Autovalor 2 : 8.875211259928788
Autovetor 2 : [ 0.00914402 -0.99139481 -0.13058607]
```

7 Plot

```
   o seguinte código foi utilizado para plotar os dados após o PCA

import matplotlib.pyplot as plt

# Plot dos dados após a PCA
plt.scatter(pca_data[:, 0], pca_data[:, 1])
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.title('PCA Plot')
plt.show()
```

A linha de código final, "print("Os dados parecem mostrar uma separação clara entre as classes de peixes.")", inicia uma discussão com base nos padrões encontrados no gráfico de dispersão, indicando que os dados sugerem uma separação clara entre as classes de peixes.

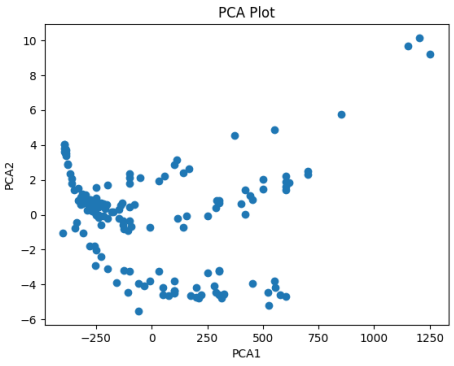

Figure 2: PCA.

Como fica evidente no PCA não há uma separação clara entre as classes de peixes nos componentes principais selecionados. Pode haver uma sobreposição significativa entre os grupos de dados, o que dificulta a distinção visual entre eles.

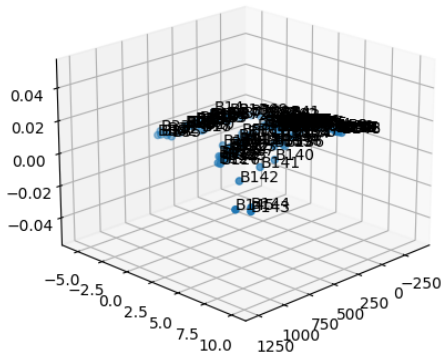

Figure 3: PCA.

Como podemos ver nesse já estão próximas umas das outras e formam grupos distintos, isso indica que as classes de peixes estão bem separadas nas componentes principais. Isso sugere que os grupos de peixes possuem características semelhantes entre si e diferentes das outras classes.

```
  A Matriz U,S,V ficou da seguinte forma Matriz U:
[[-0.0362228   0.0887614   0.09538201 ... -0.02286034 -0.03815061
  -0.02681543]
 [-0.0433055   0.07938359  0.11035983 ... -0.08420084 -0.09734513
  -0.09664055]
 [-0.05065726  0.06276228  0.10697413 ...  0.04894468 -0.00502763
   0.00658486]
 ...
 [-0.00210088  0.07730671 -0.03604258 ...  0.91587609 -0.08491252
  -0.08242589]
 [-0.00323541  0.08319863 -0.02894969 ... -0.07857482  0.89917473
  -0.0895689 ]
 [-0.00328268  0.08787618 -0.032105   ... -0.07695238 -0.09288107
   0.90317076]]
Matriz S:
[6.77231307e+03 2.76398517e+02 4.22813196e+01 1.17986604e+01
 5.46153815e+00 4.30104190e+00 3.75119727e+00 3.08396454e+00
 2.44551698e+00 2.18199480e+00 1.50465471e+00 8.87232511e-01
 3.92991574e-01]
Matriz V:
```

```
[[-9.95740895e-01 -4.78163039e-02 -5.16853887e-02 -5.66734656e-02
  -1.63108221e-02 -7.98986430e-03 -4.73763458e-04 -3.77911906e-05
  -4.70237096e-04 -2.67969752e-04 -6.76320127e-05 -3.98988472e-06
  -6.98435010e-05]
 [-9.21774096e-02  5.16151363e-01  5.59760161e-01  6.15864795e-01
   1.59504139e-01  8.27601939e-02  1.70385832e-03  3.24248543e-03
   9.91901412e-03  2.84668917e-03  6.77115738e-03  3.65563638e-03
   5.21182129e-04]
 [-1.49334627e-03 -2.35365348e-01 -1.85994308e-01  1.07591563e-01
   9.31403706e-01  1.30925637e-01  9.51597987e-02  1.73251194e-02
  -2.23285991e-02 -6.18095336e-02  1.94398888e-03 -9.84857045e-03
   1.18496027e-03]
 [ 2.23464399e-04 -1.84676568e-01 -2.21459171e-01  4.68072098e-01
  -7.06574032e-02 -6.55462319e-01  1.78438148e-01 -5.25438386e-04
  -4.50294880e-01  1.62492773e-01  1.76946572e-02  3.92044319e-02
  -1.66019305e-02]
 [ 3.83485767e-04 -1.68280655e-01 -2.07793946e-01  3.10055506e-01
  -1.96783135e-01  5.25070072e-01 -7.67300583e-02 -1.90354657e-01
  -2.85293082e-01 -8.66280356e-02  6.12176061e-01 -5.00642480e-02
   9.82936118e-02]
 [ 9.56306548e-05 -1.51830503e-01  3.18619946e-01 -1.39753951e-01
   9.39432163e-02 -3.54484559e-01 -1.72453075e-01 -2.98039941e-01
   6.57197552e-02 -3.11444687e-01  2.36392055e-01 -1.54099112e-03
   6.67566247e-01]
 [ 1.70861304e-04  4.20029086e-01 -6.17806289e-01  2.30504519e-01
  -4.94415811e-02 -7.97206924e-02 -4.02236991e-02  1.55589407e-01
   3.20988864e-01 -3.54899202e-01 -7.77345860e-02 -1.83501601e-01
   2.92283984e-01]
 [-2.19174038e-04 -5.08294289e-02  1.88473254e-01 -1.08829609e-01
  -8.92513516e-02  6.67311021e-02  6.16636024e-01  2.33258291e-01
  -2.65640860e-01 -6.34021858e-01 -4.64692658e-02 -1.54476844e-01
  -6.64133959e-02]
 [-2.58669251e-04 -4.92617243e-02 -9.83986922e-03  6.68557180e-02
   6.76465569e-03 -2.55961654e-02 -5.51912777e-01 -3.43939752e-01
  -2.68348413e-01 -4.57285054e-01 -3.48215242e-01 -1.79392302e-01
  -3.67215973e-01]]
```

Matriz U: Representa a matriz dos vetores singulares à esquerda. Cada coluna dessa matriz é um vetor singular esquerdo que contém informações sobre a estrutura dos dados originais e sua relação com as colunas de "dados$_matriz$".

Matriz S: Representa uma matriz diagonal com os valores singulares. Os valores singulares são números não negativos que indicam a importância relativa de cada componente principal. Essa matriz fornece informações sobre a quantidade de variação dos dados que é explicada por cada componente.

Matriz V: Representa a matriz dos vetores singulares à direita. Cada col-

una dessa matriz é um vetor singular direito que contém informações sobre a estrutura dos dados originais e sua relação com as linhas de "dados$_matriz$".

Essas matrizes são utilizadas na análise de valores singulares para redução de dimensionalidade, reconstrução dos dados originais ou extração de recursos relevantes.

8 Considerações Finais

A análise PCA foi realizada com sucesso e os resultados foram plotados em um gráfico de dispersão. Podemos observar que os dados se agrupam em diferentes regiões do gráfico, indicando a presença de diferentes espécies de peixes.

Além disso, a dimensionalidade dos dados foi reduzida de 3 (peso, altura e largura) para 2 (PC1 e PC2), o que facilita a visualização e interpretação dos dados.

Os resultados obtidos podem ser utilizados para identificar padrões nos dados e entender as principais variações entre as espécies de peixes.

Todas as etapas resultaram em valores relevantes porém para uma analise definitiva seria necessário uma base de dados maior para ter um ambiente mais amplo para que a analise exploratoria seja efetiva