FACULDADE DE TECNOLOGIA DA BAIXADA SANTISTA RUBENS
LARA
CIÊNCIA DE DADOS
DISCIPLINA: ÁLGEBRA LINEAR
DOCENTE: ALEXANDRE GARCIA DE OLIVEIRA

ANÁLISE PCA

PEDRO JORGE DE SOUZA COLOMBRINO
HELENA VICTÓRIA DOS SANTOS BARBOZA
MARIO AMBROSIO DE SOUZA NETO

SANTOS, NOVEMBRO DE 2023

1 INTRODUÇÃO

Nesse relatório, será descrita uma análise de componentes principais(PCA) dos dados. A análise de componentes principais ou PCA é uma técnica multivariada que permite indentificar padrões e estruturas nos dados, reduzindo sua dimensionalidade e facilitando a interpretação.

2 BANCO DE DADOS

O conjunto de dados escolhido para este estudo foi adquirido por meio do site Kaggle.com. Este arquivo consiste em 16.130 imagens de alta qualidade, com uma resolução de 512×512 pixels. Ele está dividido em três categorias: Gato, Cachorro e Animais Selvagens, sendo que cada categoria contém aproximadamente 5.000 imagens. No entanto, para a análise deste estudo, serão utilizadas apenas as pastas referentes a Gato e Cachorro.

3 CODIFICAÇÃO

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from PIL import Image
import time

# Inicia a contagem do tempo de execução
start_time = time.time()

# Carregamento de imagens de cachorros e gatos a partir dos
↪  diretórios especificados
def load_images(folder):
    images = []
    for filename in os.listdir(folder):
        img = Image.open(os.path.join(folder, filename))
        if img is not None:
            images.append(img)
    return images

# Carregamento de imagens de cachorros
```

```
dogs = load_images('D:\\PCA GARCIA\\dog')
# Carregamento de imagens de gatos
cats = load_images('D:\\PCA GARCIA\\cat')

# Pré-processamento das imagens: redimensionamento, conversão para tons de cinza e achatar as imagens
def process_images(image_list):
    processed = []
    for img in image_list:
        img = img.resize((128, 128))
        img = img.convert('L')
        img = np.array(img).flatten()
        processed.append(img)
    return processed

# Pré-processamento das imagens de cachorros
dogs = process_images(dogs)
# Pré-processamento das imagens de gatos
cats = process_images(cats)

# Criação de DataFrames pandas a partir das imagens processadas
df_dogs = pd.DataFrame(dogs)
df_dogs['label'] = 'dog'
df_cats = pd.DataFrame(cats)
df_cats['label'] = 'cat'
df = pd.concat([df_dogs, df_cats])

# Exibe o tamanho da matriz antes da classificação
print("O tamanho da matriz antes da classificação é: ", df.shape)

# Divisão dos dados em conjuntos de treinamento e teste
print("Dividindo os dados em conjuntos de treinamento e teste.")
X = df.drop('label', axis=1)
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Normalização dos dados
print("Normalizando os dados.")
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

```

```
67  # Cálculo dos autovalores e autovetores da matriz de
    ↪ covariância
68  print("Calculando os autovalores e autovetores da matriz.")
69  eigenvalues, eigenvectors = np.linalg.eig(X_train.T @
    ↪ X_train)
70  print('Autovalores: ', eigenvalues)
71  print('Autovetores: ', eigenvectors)
72
73  # Aplicação do PCA (Análise de Componentes Principais) aos
    ↪ dados
74  print("Aplicando PCA aos dados.")
75  pca = PCA(n_components=3)
76  X_train_pca = pca.fit_transform(X_train)
77  X_test_pca = pca.transform(X_test)
78
79  # Exibe o tamanho da matriz após a classificação
80  print("O tamanho da matriz após a classificação é: ",
    ↪ X_train_pca.shape)
81
82  # Treinamento do classificador Support Vector Machine (SVM)
    ↪ nos dados de treinamento
83  print("Treinando o classificador SVC nos dados de
    ↪ treinamento.")
84  svc = SVC()
85  svc.fit(X_train_pca, y_train)
86
87  # Faz previsões nos dados de teste
88  print("Fazendo previsões nos dados de teste.")
89  y_pred = svc.predict(X_test_pca)
90
91  # Cálculo da acurácia da classificação
92  print("Calculando a acurácia da classificação.")
93  accuracy = accuracy_score(y_test, y_pred)
94  print('Acurácia da classificação: ', accuracy)
95
96  # Plotagem dos gráficos tridimensionais antes e depois da
    ↪ aplicação do PCA
97  print("Plotando os gráficos.")
98  fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 8),
    ↪ subplot_kw={'projection': '3d'})
99
100 # Gráfico antes da classificação
101 ax1.set_xlabel('Feature 1', fontsize=15)
102 ax1.set_ylabel('Feature 2', fontsize=15)
103 ax1.set_zlabel('Feature 3', fontsize=15)
```

```
104 ax1.set_title('Dados originais - Antes da classificação',
    ↪ fontsize=20)
105
106 targets = ['dog', 'cat']
107 colors = ['b', 'r']
108 for target, color in zip(targets, colors):
109     indicesToKeep = df['label'] == target
110     ax1.scatter(df.loc[indicesToKeep, df.columns[0]],
111                 df.loc[indicesToKeep, df.columns[1]],
112                 df.loc[indicesToKeep, df.columns[2]],
113                 c=color,
114                 s=50)
115 ax1.legend(targets)
116 ax1.grid()
117
118 # Gráfico após a classificação
119 ax2.set_xlabel('Principal Component 1', fontsize=15)
120 ax2.set_ylabel('Principal Component 2', fontsize=15)
121 ax2.set_zlabel('Principal Component 3', fontsize=15)
122 ax2.set_title('3 component PCA - Após a classificação',
    ↪ fontsize=20)
123
124 df_pca = pd.DataFrame(data=X_train_pca, columns=['pc1',
    ↪ 'pc2', 'pc3'])
125 df_pca['label'] = y_train.values
126
127 for target, color in zip(targets, colors):
128     indicesToKeep = df_pca['label'] == target
129     ax2.scatter(df_pca.loc[indicesToKeep, 'pc1'],
130                 df_pca.loc[indicesToKeep, 'pc2'],
131                 df_pca.loc[indicesToKeep, 'pc3'],
132                 c=color,
133                 s=50)
134 ax2.legend(targets)
135 ax2.grid()
136
137 # Exibe os gráficos
138 plt.show()
139
140 # Exibe o tempo total de execução
141 print("--- %s seconds ---" % (time.time() - start_time))
```

4 RESULTADOS

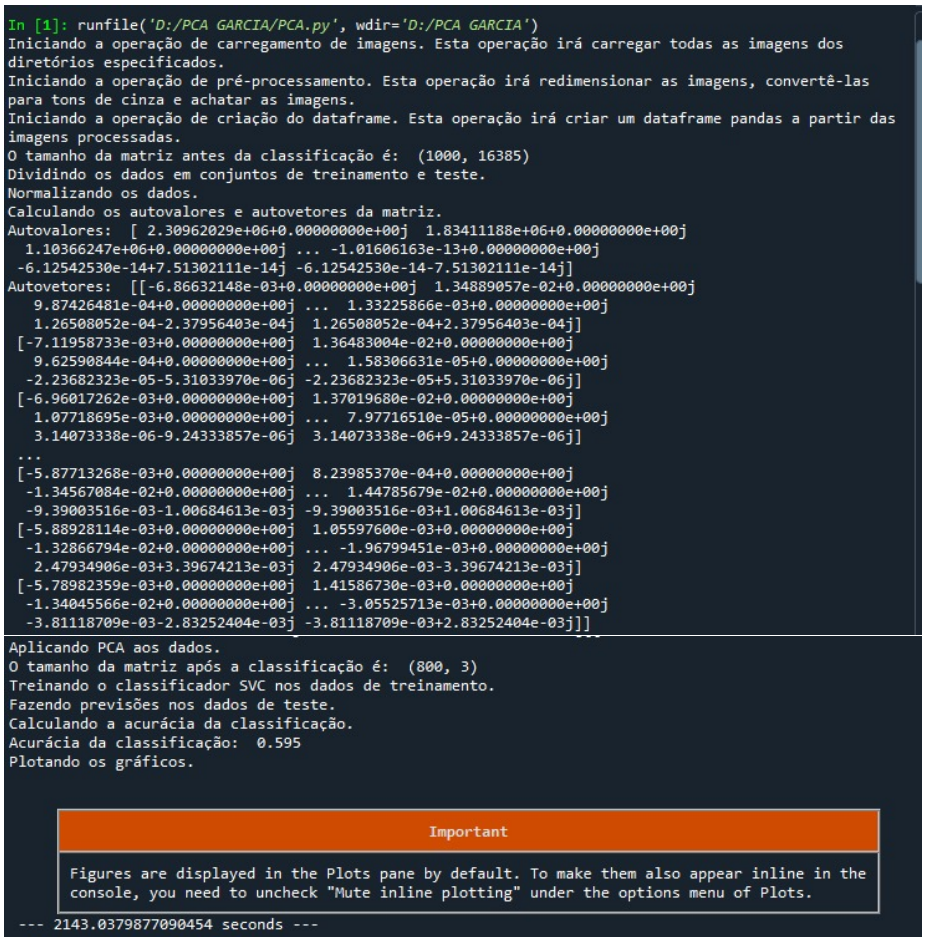

Figura 1: Resultados do código

O código em Python realiza uma Análise de Componentes Principais (PCA) em um conjunto de imagens de cachorros e gatos, utilizando bibliotecas como NumPy, Pandas e Matplotlib. As etapas incluem o carregamento e processamento das imagens, organização dos dados em DataFrames, divisão em conjuntos de treinamento e teste, normalização dos dados e o cálculo de autovalores e autovetores da matriz de covariância.

O conjunto de dados consiste em 1000 imagens, cada uma com 16385 pixels. As imagens passam por pré-processamento, redimensionamento, conversão para tons de cinza e achatamento, sendo transformadas em um DataFrame do Pandas. O conjunto de treinamento é usado para treinar o modelo PCA, enquanto o conjunto de teste avalia o desempenho do modelo.

A normalização é crucial para garantir a correta execução da PCA, e os autovalores e autovetores da matriz de dados são calculados. A PCA reduz a dimensionalidade de 16385 para 3, representando cada imagem como uma combinação linear de três componentes principais.

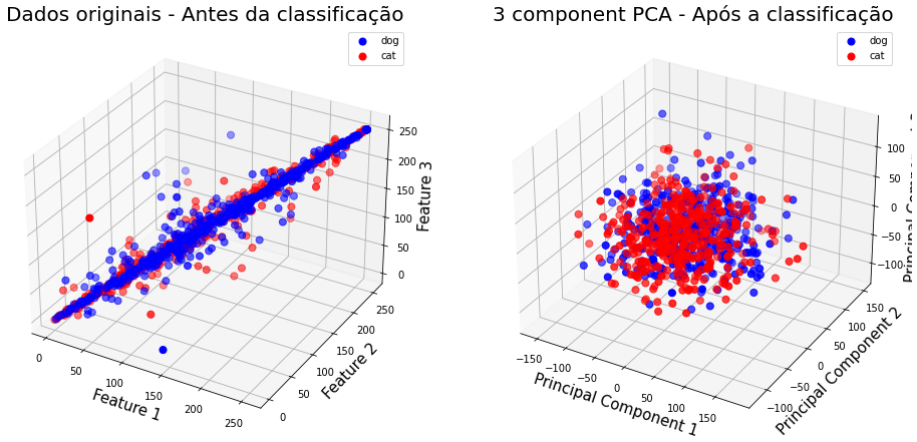

Figura 2: Separação de classes por PCA

Na fase de classificação, o código utiliza um classificador de suporte vetorial (SVC) para categorizar as imagens em duas classes: gato e cachorro. O SVC é um modelo de aprendizado de máquina supervisionado comum em tarefas de classificação de imagens.

A visualização dos dados é feita por meio de um gráfico de dispersão, onde pontos azuis representam imagens de cachorros e pontos vermelhos representam imagens de gatos. O gráfico revela uma sobreposição significativa entre os grupos, indicando características bastante semelhantes e dificuldade na distinção visual.

5 CONSIDERAÇÕES FINAIS

Em conclusão, a Análise de Componentes Principais (PCA) foi realizada com sucesso e os resultados plotados em um gráfico de dispersão. Apesar do sucesso na redução de dimensionalidade e classificação, a visualização revela uma sobreposição significativa entre as classes, indicando características semelhantes.