Análise Exploratória de Dados

Professor: Ricardo Coelho
Equipe:
Sara Martins
Kelvin Oliveira
Nayla Moraes

Projeto de Análise do dataset sobre a Felicidade Mundial

Trabalhamos para identificar possíveis fatores que afetem o índice de felicidade (score) nos países.

- ## Análise Geral:

Fizemos o nosso código em Python. Iniciamos no Google Colab, pois é mais fácil de organizar as partes de cada integrante e de colocar textos para informações mais pontuais. Após, foi gerado um link que conectou os códigos do IDE do Colab para a Plataforma do Github, onde foi dividido no repositório da equipe nas seguintes pastas:

- **.devcontainer :** configura um ambiente de desenvolvimento isolado que funciona independentemente do sistema operacional dos colaboradores do código;

- .**gitignore** : lista que invalida certas páginas ou pastas de serem rastreadas ou enviadas para o repositório do Github;

- **.vscode :** configurações específicas do vscode, como a formatação, o path;

- **2019.csv :** local onde fica o dataset utilizado para esse projeto;

- **Felicidade_mundial.ipynb**

- **app.py :** onde está a base do nosso código

- **Requirements.txt** : lista com todas as bibliotecas utilizadas
  - **Streamlit**: biblioteca Python que torna códigos e scripts em páginas web interativas para a análise e compreensão dos dados;
  - **Pandas**: análise e manipulação de dados tabulares;
  - **Numpy**: suporte matemático, de arrays, matrizes e de funções;
  - **Matplotlib**: biblioteca para a visualização por meio de gráficos estáticos, interativos e animados;
  - **Seaborn**: biblioteca para a visualização de dados estatísticos mais elegantes;

- **Scipy**: biblioteca para funções matemáticas, estatísticas e científicas;
- **Ploly**: biblioteca para visualização interativa de dados por meio de gráficos bonitos e responsivos;
- **Pycountry**: biblioteca para dados fáceis sobre outros países;

Análise do Código:

Instalação de bibliotecas importantes e a importação de funções de determinadas bibliotecas, além de importar as bibliotecas pela abreviação, para facilitar a organização do código e não precisar escrever todo o nome, poluindo o código

```
!pip install pycountry-convert
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pycountry_convert as pc
import numpy as np
from scipy import stats
```

Após a instalação (pip install …) e importação, aparecerá no terminal se foi concluído ou não o processo.

Aqui estamos importando pelo link web o dataset que iremos utilizar, mas também pode-se baixar o arquivo e colocar na mesma página que estão seus outros arquivos do projeto para que, ao longo do código, o link possa ser chamado.

```
url =
"https://raw.githubusercontent.com/kevin-oliveira178/Felicidade_Mudial/refs/heads/main/2019.csv"
 df = pd.read_csv(url)

 colunas = list(df.columns)
```

O df está lendo o link do csv, que significa “valores separados por vírgulas” e os nomes das colunas do dataset estão sendo convertidas para uma lista Python.

Assim como dito no comentário dessa parte do código, uma nova coluna estará sendo adicionada a lista já existente. Essa coluna classifica numericamente o “grau de felicidade”, transformando-o em uma variável qualitativa ordinal em que valores, até certo limite, são considerados pertencentes de um dos três tercis criados, dependendo se ultrapassam ou estão dentro dos valores limitantes definidos. Como serão classificados por 3 números, foram criados três tercis, que dividem o todo em três partes.

```
#criação das colunas adicionais
```

```
#coluna Score_Category
score = df['Score')
# calculo dos tercis
tercis = sc.quantile([0.3333, 0.6667])
tercil1 = tercis[0.3333]
tercil2 = tercis[0.6667]
```

A coluna score foi chamada e foi calculado os valores que limitam os tercis de forma aproximada.

Aqui, a função categorizar será definida com o parâmetro score. Nessa função, uma nova coluna será criada e ela lerá os valores apresentados no dataset. Caso o valor i esteja no primeiro tercil, até 33,3%, este valor será tido como “baixo”, caso o valor pertença tanto ao primeiro quanto ao segundo quartil, será adicionado o valor “Médio”, mas se o valor for maior do que o limitante superior do segundo tercil, então este será adicionado na coluna com o valor “Alto”. No final, está categorizando a coluna score.

```
#adicionando a coluna

def categorizar_score(score):
  new_colum = []
  for i in score:
      if i <= tercil1:
          new_colum.append("Baixo")
      elif i > tercil1 and i <= tercil2:
          new_colum.append("Médio")
      elif i > tercil2:
          new_colum.append("alto")
  return new_colum

df["Score_Category"]= categorizar_score(df["Score"])
```

Essa tabela de frequência que será criada acessa a coluna “Score Category” e conta quantos valores “Alto”, “Médio” e “Baixo” ocorreram nessa coluna.

```
#criação da tab_Score_category de frequência da variável
Score_Category
tab_Score_category = df["Score_Category"].value_counts()
```

Outras funções estão sendo criadas para podermos agrupar melhor os dados. Dessa forma, iremos criar uma coluna que identifica o continente de cada país de cada linha do dataset e terá uma função que vai descobrir o continente desse país. Após conseguir o nome do continente, será pedido por meio do código que o nome completo do continente seja substituído pelo nome completo do respectivo continente, advindo da sigla do nome. Caso o país seja inválido ou desconhecido, o valor retornado serpa “Unknown” = “desconhecido”.

```
#definindo a função que cria a coluna de continentes

def add_continent_column(df, country_col):
    def get_continent(country):
        try:
            country_code =
pc.country_name_to_country_alpha2(country, cn_name_format="default")
            continent_code =
pc.country_alpha2_to_continent_code(country_code)
            continent_name = {
                'AF': 'Africa',
                'NA': 'North America',
                'OC': 'Oceania',
                'AN': 'Antarctica',
                'AS': 'Asia',
                'EU': 'Europe',
                'SA': 'South America'
            }.get(continent_code, 'Unknown')
            return continent_name
        except:
            return 'Unknown'

    df['continent'] = df[country_col].apply(get_continent)
    return df
```

Assim, a função get_continent é aplicada a todo valor da coluna, retornando valores adicionadas a coluna continente.

```
Atualização do próprio dataframe por meio da função
#chamando a função
df = add_continent_column(df, 'Country or region')
```

Parte 1:

Nesse bloco do código, está sendo calculadas as medidas de tendência central, dispersão e inúmeros outros coeficientes, todas as fórmulas aplicadas ao score, que seria o grau de felicidade.

```
#descrição da variável score
score = df["Score"]

# Descrição da variável score
media = score.mean()
moda = score.mode()
mediana = score.median()

# medidas de dispersão
amplitude = score.max() - score.min()
desvio_medio = np.mean(np.abs(score - score.mean()))
variancia = score.var(ddof=1)          # ddof=1 → variância amostral
desvio_padrao = score.std(ddof=1)
coef_variacao = (desvio_padrao / score.mean()) * 100
iqr = score.quantile(0.75) - score.quantile(0.25)
quartil_1 = score.quantile(0.25)
quartil_2 = score.quantile(0.5)
quartil_3 = score.quantile(0.75)
```

Aqui está retornando todas as medidas e os resultados dos cálculos de cada medida

```
print('\n média \n', media,
      '\nmoda\n',moda,
      '\nmediana\n',mediana,
      '\namplitude total\n',amplitude,
      '\ndesvio padrão/n',desvio_medio,
      '\nvariancia\n',variancia,
      '\ndesvio padrão',desvio_padrao,
      '\ncoef de variação\n',coef_variacao,
      '\namplitude interquartilica\n',iqr,
      '\nprimeiro quartil\n',quartil_1,
      '\nsegundo quartil\n',quartil_2,
      '\nterceiro quartil\n',quartil_3)
```

Aqui temos a análise descritiva e os parâmetros da criação de um gráfico.

No início está evidente a criação da paleta de cores própria, a cor do fundo do gráfico, a cor do texto.

```
# Paleta personalizada
colors = ["#ff6b6b", "#feca57", "#48dbfb", "#1dd1a1", "#5f27cd"]
background_color = "#ffffff"
```

```
    text_color = "#222f3e"
    accent_color = "#10ac84"
```

Uma das fórmulas para definir o número de classes que o gráfico terá, de modo que esteja bem organizado e com uma quantidade de classes útil para a observação e condizente com o tamanho da amostra analisada é o Método de Sturgees.

```
    # Função para definir número de classes pelo método de Sturges
    def sturges_classes(series):
        n = len(series)
        return int(np.ceil(1 + 3.3 * np.log10(n)))
```

Serão criadas as classes por meio do cálculo do valor mínimo e máximo de cada intervalo de classe.

```
    # Etapa 1: Criar classes (agrupamento)
    k = sturges_classes(score)
    min_val = score.min()
    max_val = score.max()
```

Cria os intervalos entre os valores mínimos e máximos de cada intervalo individual, contendo valores entre esses intervalos que se enquadram com características de todo o intervalo

```
    bins = np.linspace(min_val, max_val, k + 1)
```

Criação do dataframe de frequência com valores limitantes definidos. Além disso, tem o cálculo da amplitude e do ponto médio, dois pontos importantíssimos em uma tabela de frequência, visto que a maioria dos cálculos estatísticos giram em torno desses dois valores.

```
    # Etapa 2: Criar tabela de frequência
    df_freq = pd.DataFrame({
        'classe_inferior': bins[:-1],
        'classe_superior': bins[1:]
    })
    df_freq['amplitude'] = df_freq['classe_superior'] -
df_freq['classe_inferior']
    df_freq['ponto_medio'] = (df_freq['classe_inferior'] +
df_freq['classe_superior']) / 2
```

Contabiliza a quantidade de valores presentes em cada intervalo, evidenciando a frequência de cada classe. No df da frequência, corta em intervalos e inclui o menor valor(limitante) e mantém a ordem do intervalo, extraindo apenas o números de ocorrências.

```
        # Contar quantos valores caem em cada intervalo (classe)
        df_freq['frequencia'] = pd.cut(score, bins=bins,
include_lowest=True).value_counts(sort=False).values
        df_freq['fac'] = df_freq['frequencia'].cumsum()
```

```
    # Etapa 3: Média agrupada
    media_agrupada = (df_freq['frequencia'] *
df_freq['ponto_medio']).sum() / df_freq['frequencia'].sum()

    # Etapa 4: Mediana agrupada
    N = df_freq['frequencia'].sum()
    N2 = N / 2
    classe_mediana_idx = df_freq[df_freq['fac'] >= N2].index[0]
    L = df_freq.loc[classe_mediana_idx, 'classe_inferior']
    F = df_freq.loc[classe_mediana_idx - 1, 'fac'] if classe_mediana_idx
> 0 else 0
    f_m = df_freq.loc[classe_mediana_idx, 'frequencia']
    h = df_freq.loc[classe_mediana_idx, 'amplitude']
    mediana_agrupada = L + ((N2 - F) / f_m) * h

    # Etapa 5: Moda agrupada (Czuber)
    classe_modal_idx = df_freq['frequencia'].idxmax()
    f1 = df_freq.loc[classe_modal_idx, 'frequencia']
    f0 = df_freq.loc[classe_modal_idx - 1, 'frequencia'] if
classe_modal_idx > 0 else 0
    f2 = df_freq.loc[classe_modal_idx + 1, 'frequencia'] if
classe_modal_idx < len(df_freq) - 1 else 0
    L_modal = df_freq.loc[classe_modal_idx, 'classe_inferior']
    h_modal = df_freq.loc[classe_modal_idx, 'amplitude']
    moda_agrupada = L_modal + ((f1 - f0) / ((f1 - f0) + (f1 - f2))) *
h_modal if (f1 - f0 + f1 - f2) != 0 else np.nan

    # Etapa 6: Plotar o histograma com linhas das medidas
    plt.figure(figsize=(10, 5))
```

```
    sns.set_style("whitegrid", {"axes.facecolor": background_color})
    sns.set_palette(sns.color_palette(colors))

    # Histograma com os mesmos bins usados no agrupamento
    sns.histplot(score, bins=bins, edgecolor='black', alpha=0.8)

    # Linhas verticais das medidas agrupadas
    plt.axvline(media_agrupada, color='#ff6b6b', linestyle='--',
linewidth=2, label=f'Média agrupada: {media_agrupada:.2f}')
    plt.axvline(mediana_agrupada, color='#1dd1a1', linestyle='-.',
linewidth=2, label=f'Mediana agrupada: {mediana_agrupada:.2f}')
    if not np.isnan(moda_agrupada):
        plt.axvline(moda_agrupada, color='#5f27cd', linestyle=':',
linewidth=2, label=f'Moda agrupada: {moda_agrupada:.2f}')

    # Estilo e legendas
    plt.title(f'Histograma da Felicidade (Score) com Tendência Central
(Dados Agrupados)', color=text_color, fontsize=14)
    plt.xlabel('Score de Felicidade', color=accent_color)
    plt.ylabel('Frequência', color=accent_color)
    plt.xticks(color=text_color)
    plt.yticks(color=text_color)
    plt.legend()
    plt.gca().spines['top'].set_visible(False)
    plt.gca().spines['right'].set_visible(False)
    plt.gca().spines['left'].set_color(text_color)
    plt.gca().spines['bottom'].set_color(text_color)
    plt.grid(True, axis='y', linestyle='--', alpha=0.7)
    plt.gcf().set_facecolor(background_color)
    plt.show()
```