Atividade 3 - Aprendizado de Máquina

AUTHOR
Rennan Dalla Guimarães

1.1 Setup e carregamento

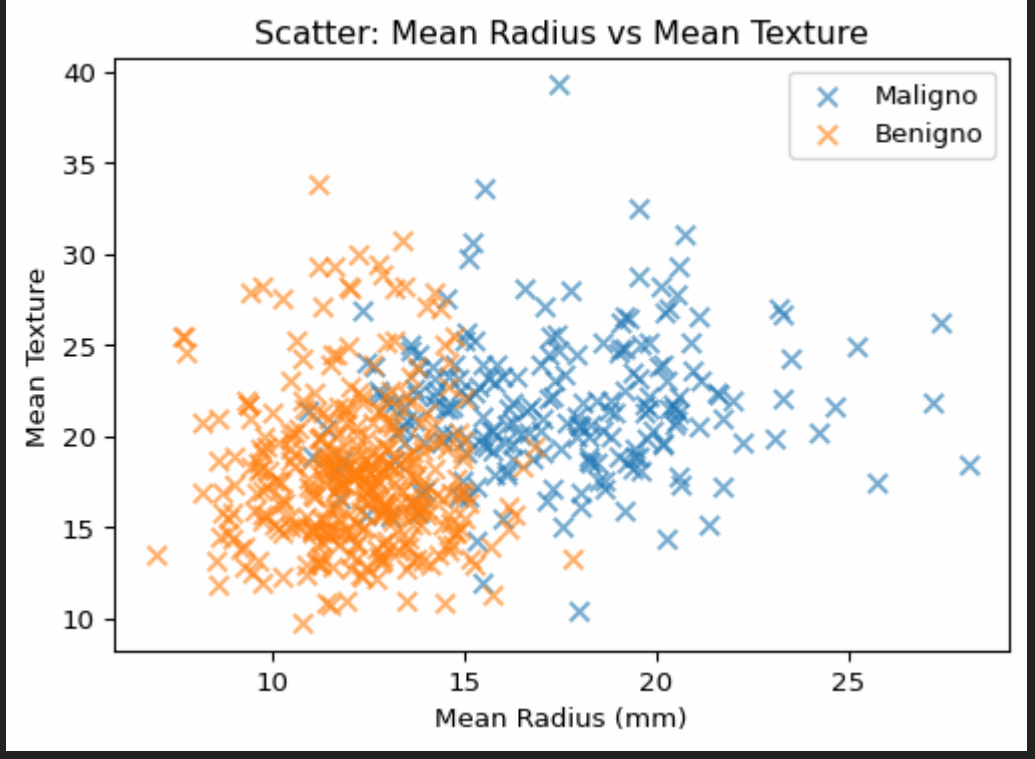
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer

# Carrega o dataset
data = load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df["diagnosis"] = data.target   # 0 = maligno, 1 = benigno

df.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

5 rows × 31 columns

1.2 Gráfico de dispersão

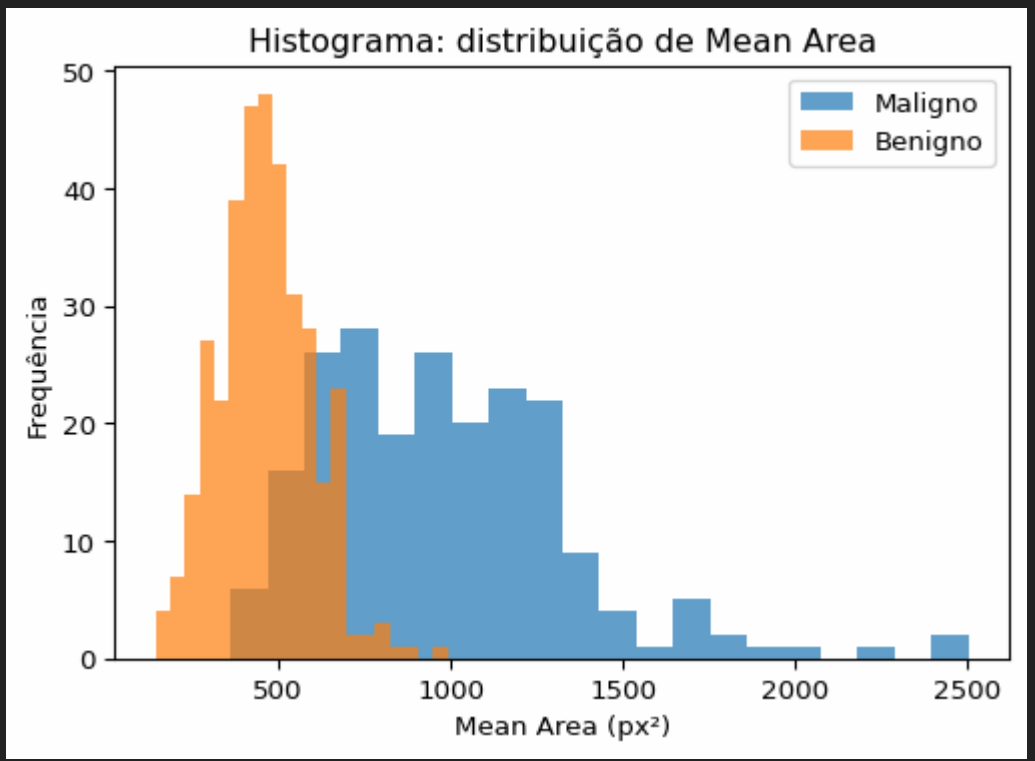
```
fig, ax = plt.subplots(figsize=(6,4))
colors = {0: "tab:orange", 1: "tab:red"}
labels = {0: "Maligno", 1: "Benigno"}

for label, grp in df.groupby("diagnosis"):
    ax.scatter(grp["mean radius"], grp["mean texture"],
               alpha=0.6, label=labels[label],
               marker="x", s=50)

ax.set_xlabel("Mean Radius (mm)")
ax.set_ylabel("Mean Texture")
ax.set_title("Scatter: Mean Radius vs Mean Texture")
ax.legend()
plt.show()
```



Observação: lesões malignas concentram-se em raios **≥ 15 mm** e texturas **≥ 20**, sugerindo forte poder discriminatório desses atributos.

1.3 Histograma sobreposto – *mean area*

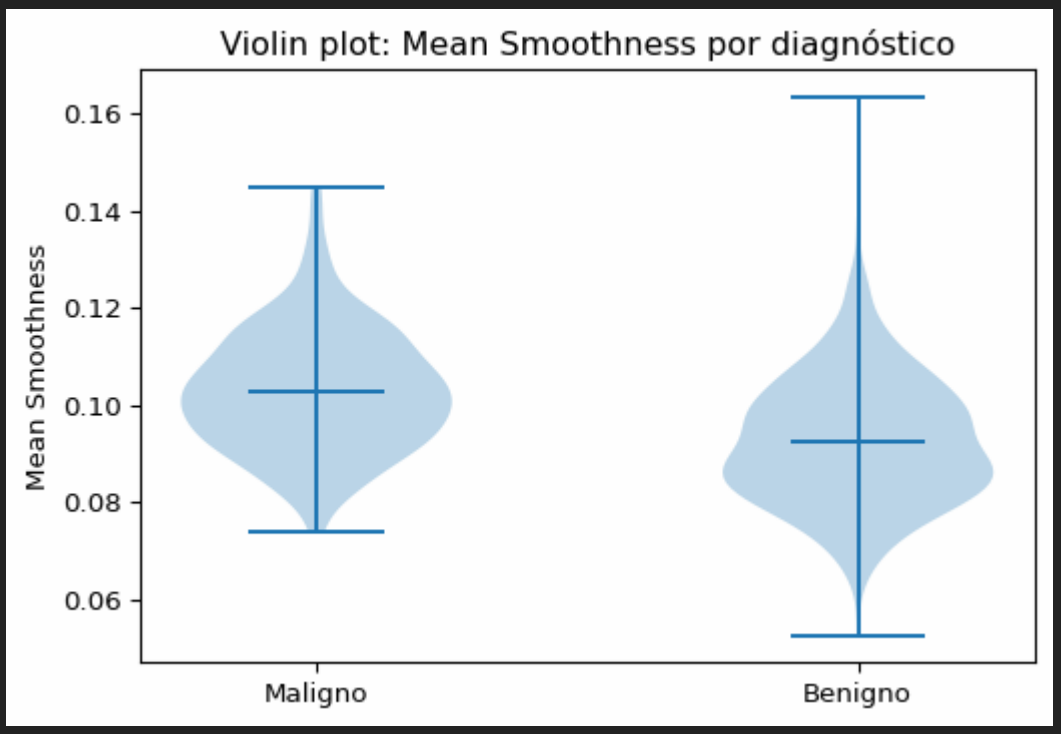
```
fig, ax = plt.subplots(figsize=(6,4))
bins = 20
ax.hist(df[df["diagnosis"]==0]["mean area"], bins=bins, alpha=0.7, lab
ax.hist(df[df["diagnosis"]==1]["mean area"], bins=bins, alpha=0.7, lab
ax.set_xlabel("Mean Area (px²)")
ax.set_ylabel("Frequência")
ax.set_title("Histograma: distribuição de Mean Area")
ax.legend()
plt.show()
```



Observação: a distribuição é bimodal; **mean area** separa bem as classes (malignos têm áreas muito maiores).

1.4 Violin plot – *mean smoothness*

```
fig, ax = plt.subplots(figsize=(6,4))
data_violin = [
    df[df["diagnosis"]==0]["mean smoothness"],
    df[df["diagnosis"]==1]["mean smoothness"],
]
ax.violinplot(data_violin, showmeans=True)
ax.set_xticks([1,2])
ax.set_xticklabels(["Maligno","Benigno"])
ax.set_ylabel("Mean Smoothness")
ax.set_title("Violin plot: Mean Smoothness por diagnóstico")
plt.show()
```



---

2 Resumo – *Python Data Visualization Essentials Guide* (Capítulos 1–3)

2.1 Capítulo 1 – *Introduction to Data Visualization*

Este primeiro capítulo funciona como um "por que estamos fazendo tudo isso?". O autor mostra que a visualização de dados não é enfeite, mas sim uma ponte entre números brutos e decisões de negócio.
Alguns pontos que me marcaram:

- **Visualização como linguagem universal.** Diferentes áreas (estatística, design e storytelling) se cruzam aqui. É quase como aprender um novo idioma: em vez de conjugar verbos, combinamos cores, formas e posições.
- **Pirâmide de prioridades.** Gostei muito da lógica *Precisão* → *Clareza* → *Estética*. É tentador começar escolhendo a paleta de cores, mas se o gráfico não for fiel aos

dados nem fizer sentido de relance, a beleza não salva.
- **Processo iterativo.** O livro defende prototipar cedo, coletar feedback rápido e refinar. Isso me lembrou da metodologia ágil no desenvolvimento de software: falhar rápido para acertar mais cedo.

2.2 Capítulo 2 – *Why Data Visualization?*

Aqui o autor responde à pergunta que muita gente faz: "Por que perder tempo fazendo gráfico se eu posso simplesmente mandar uma tabela em Excel?".
Os argumentos são convincentes:

1. **Velocidade de compreensão.** Estudos citados mostram que o cérebro reconhece padrões visuais em milissegundos, enquanto interpretar números crus leva segundos ou minutos.
2. **Retenção de informação.** Gráficos bem feitos ficam na memória. Estatística do livro: 65 % de retenção de imagens depois de 72 h contra 10 % de texto puro — ou seja, se quero que meu chefe lembre da análise na reunião de terça, é melhor caprichar na visual.
3. **Casos reais.** Gosto quando o autor apresenta exemplos do "mundo real": detecção de fraude em cartão a partir de scatter plots, dashboards que economizam milhões em logística. Dá senso de urgência e mostra que não é só teoria.

Um detalhe ético importante: gráficos podem persuadir, mas também manipular. Usar eixo truncado ou cores enganosas pode levar a conclusões erradas. O livro bate bastante nessa tecla de responsabilidade.

2.3 Capítulo 3 – *Various Data Visualization Elements and Tools*

Depois de entender o *porquê*, este capítulo entra no *como*.

- **Match gráfico-pergunta.** O autor sugere sempre começar pela pergunta ("Quero mostrar correlação, distribuição ou tendência temporal?") e só depois escolher o tipo de gráfico. Isso evita modinha ("todo mundo usando treemap, então vou usar também").
- **Hierarquia perceptual.** Posição é o canal visual mais preciso, seguida de comprimento. Área e cor vêm depois. Ótima regra para lembrar quando fico em dúvida entre usar bubble chart ou scatter simples.
- **Panorama do ecossistema Python.**
  - *Matplotlib* – canivete suíço; faz de tudo, mas exige mais código.
  - *Seaborn* – camada estatística que simplifica correlações e distribuições.
  - *Plotly/Bokeh* – quando interatividade importa (ex.: dashboards).
  - *Altair* – abordagem declarativa; ótima para prototipar rápido.

O capítulo fecha com dicas práticas: versionar scripts, separar dados de estilização e automatizar gráficos recorrentes. São conselhos valiosos para evitar aquele clássico "onde salvei a última versão do gráfico?".

> **Conexão com o trabalho prático:**
> Depois de ler esses capítulos, ficou claro por que os atributos de área e raio foram tão eficazes na análise exploratória do câncer de mama. Eles atendem aos

princípios de posição/comparação discutidos pelo autor, permitindo que diferenças saltem aos olhos sem truques visuais.