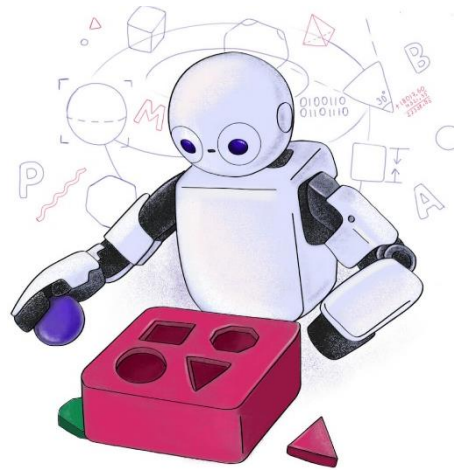


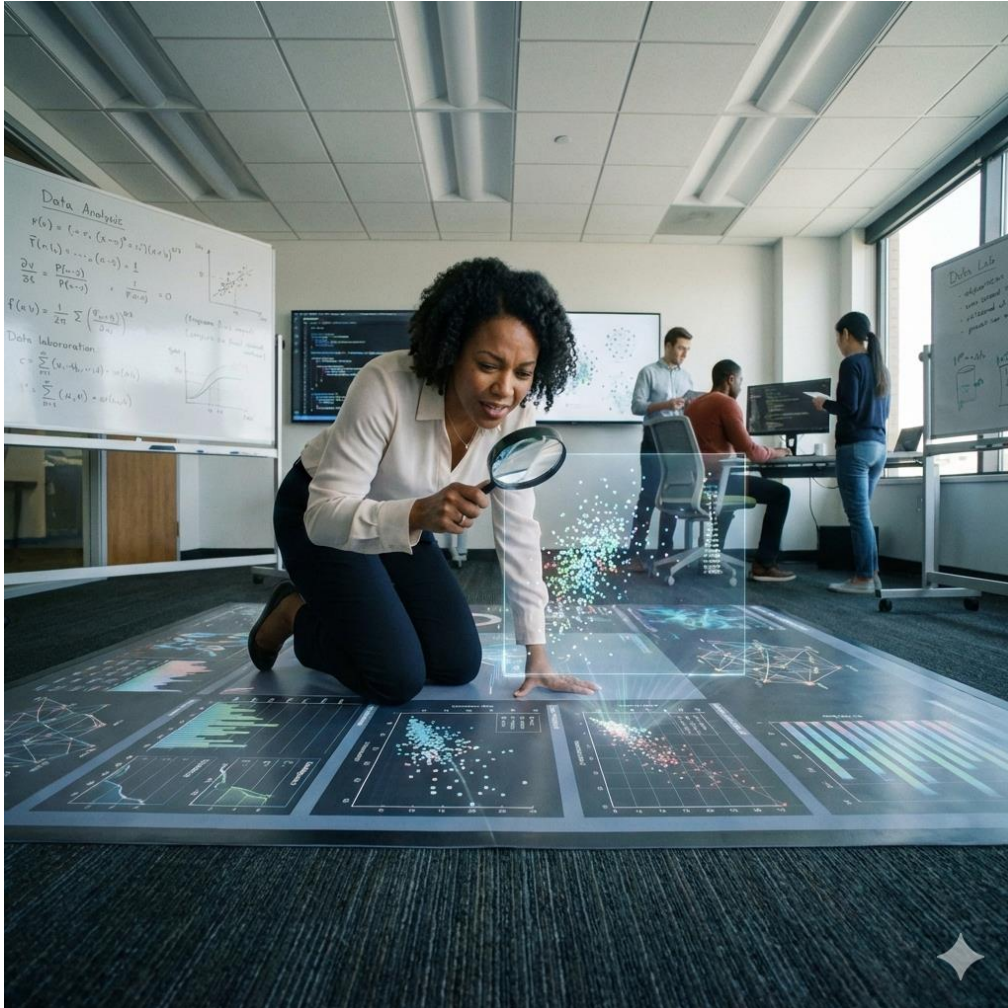
C24 - Inteligência Artificial: Análise Exploratória de Dados (EDA)





Etapa executada
antes da construção
e treinamento dos
modelos de ML.

Introdução



- É o primeiro passo em qualquer projeto de ciência de dados, incluindo ML.
- EDA nos ajudar a identificar:
 - erros óbvios nos dados,
 - compreender os padrões nos dados,
 - detectar valores discrepantes, faltantes, duplicados ou eventos anômalos,
 - encontrar relações entre as variáveis e
 - descobrir quais variáveis realmente importam.

Por que realizar análise exploratória?



- A análise exploratória é uma espécie de curadoria dos dados.
- Ela garante que o modelo de ML não aprenda com lixo.
- Se o dado de entrada é ruim, a previsão será pior ainda.

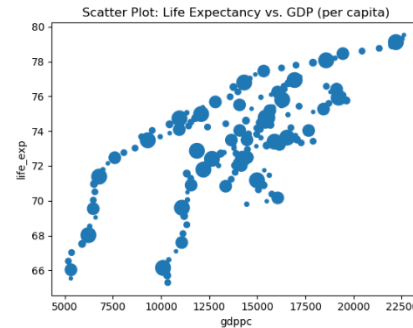
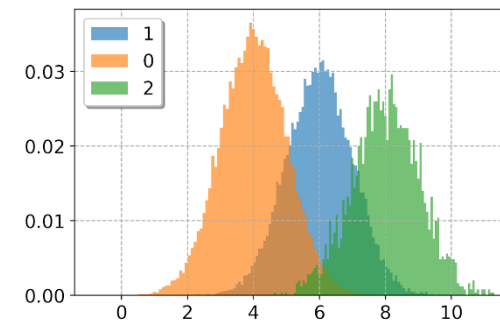
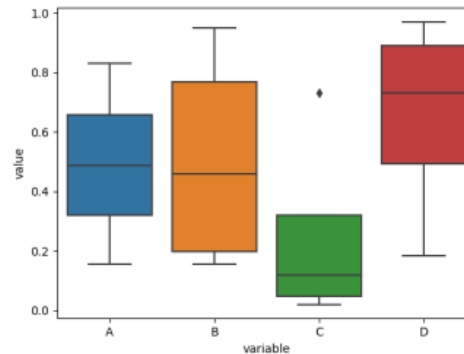
Inspeção visual e estatística

- Utilizamos ferramentas estatísticas e de visualização como

- Histogramas
- *Heatmaps* de correlação
- *Boxplots*
- Gráficos de barras
- Diagramas de dispersão

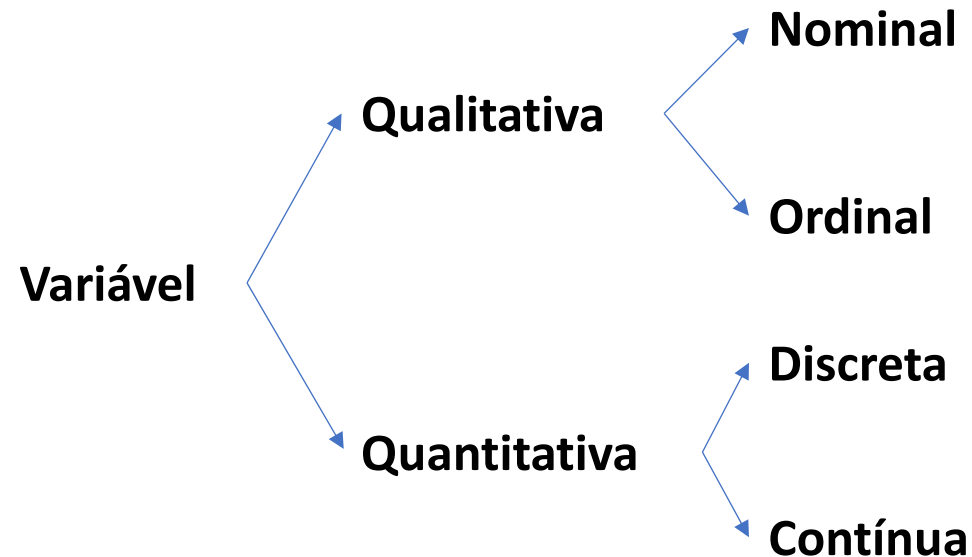
- Bibliotecas mais usadas:

- Pandas
- Numpy
- Scikit-learn
- Matplotlib
- Seaborn
- Pandas profiling



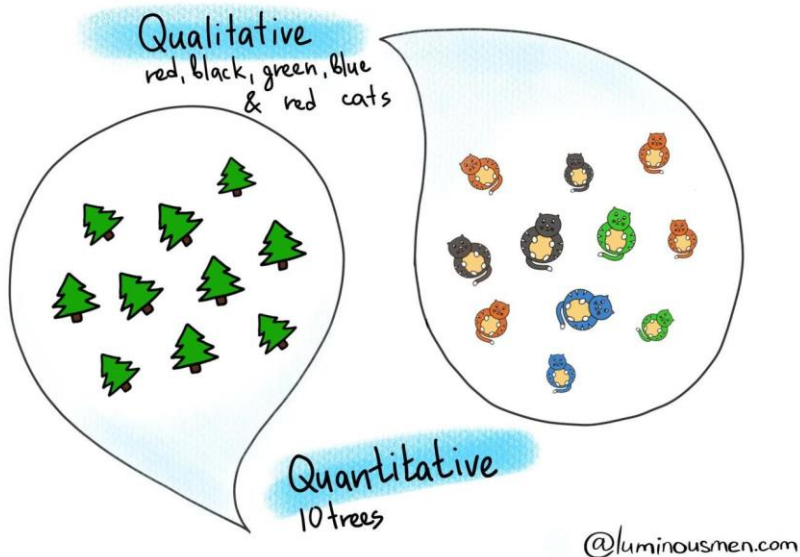
Tipo dos dados

- Antes de explorarmos os dados, precisamos entender a sua taxonomia.
- As variáveis (i.e., atributos ou rótulos) podem ser classificadas da seguinte forma:



Tipo dos dados: variáveis qualitativas (categóricas)

Qualitative and Quantitative Observations



São categorias, símbolos, nomes ou rótulos.

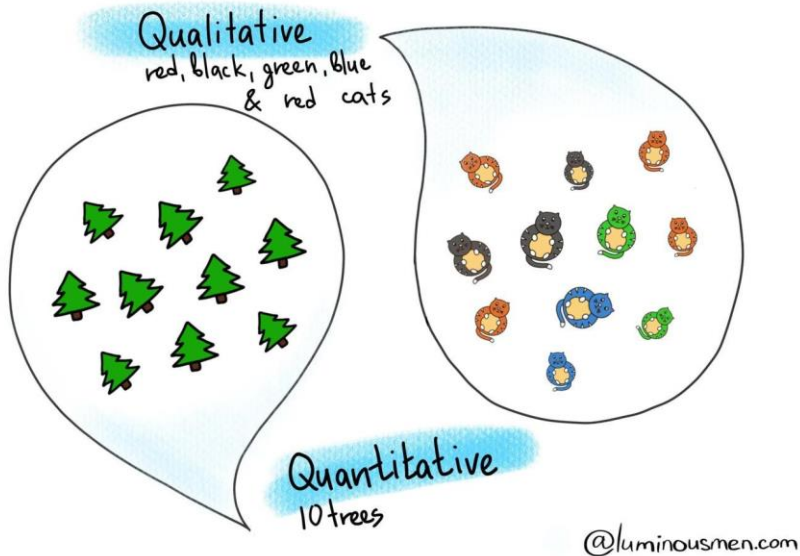
Elas descrevem uma qualidade.

Algumas podem ser ordenadas, mas operações aritméticas não são aplicáveis.

- **Nominal:** Não existe uma ordem inerente.
 - Exemplos: Sistemas Operacionais (Linux, Windows, macOS), Cores de LED, Tipos de Banco de Dados.
- **Ordinal:** Existe uma hierarquia ou ordem lógica.
 - Exemplos: Febre (baixa, média, alta), Nível de Senioridade (Junior, Pleno, Sênior), Planos de Assinatura (Free, Premium, Enterprise).

Tipo dos dados: variáveis quantitativas (numéricas)

Qualitative and Quantitative Observations



São números reais.

Podem ser ordenados e usados em operações aritméticas (e.g., média, variância).

Possuem unidade de medida.

- **Discreta:** Valores contáveis, geralmente números inteiros.
 - Exemplos: Quantidade de núcleos da CPU, número de bugs abertos no Jira, total de usuários ativos.
- **Contínua:** Valores que podem assumir qualquer número dentro de um intervalo.
 - Exemplos: Tempo de resposta de uma API, latência de rede, tamanho de um arquivo em MB, temperatura do processador.

Inspeção inicial dos dados

Inicialmente, verificamos algumas informações básicas dos dados.

Para tal, após carregarmos os dados, usamos atributos e métodos da biblioteca **pandas**.

- Dimensão dos dados: linhas x colunas
 - `df.shape`
- Tipos de dados: numéricos, categóricos, etc.
 - `df.dtypes`
- Estatísticas básicas: média, min, max, quartis, mediana, desvio padrão, etc.
 - `df.describe()`

Limpeza dos dados

Alguns problemas comuns que podemos encontrar nos dados são:

- Valores irrelevantes
- Duplicatas
- Valores faltantes
- Tipos inconsistentes (e.g., número como texto, datas quebradas)
- *Outliers*

Vamos ver como lidar com cada um deles na sequência.

Removendo valores irrelevantes

- A ideia é manter colunas que ajudem o objetivo do problema e remover as que atrapalham.
 - `df.drop(['Column 1', 'Column 2', 'Column N'], axis=1)`
- Alguns motivos para eliminar colunas: IDs, *timestamps*, coluna com valores constantes, coluna com muitos valores faltantes, colunas redundantes, etc.

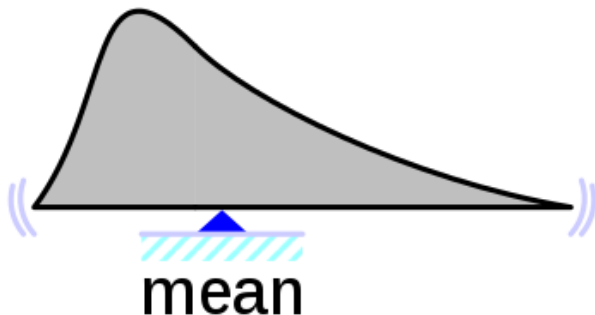
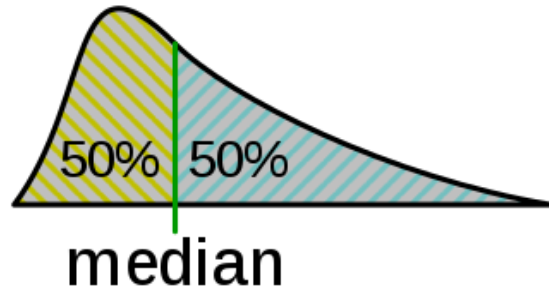
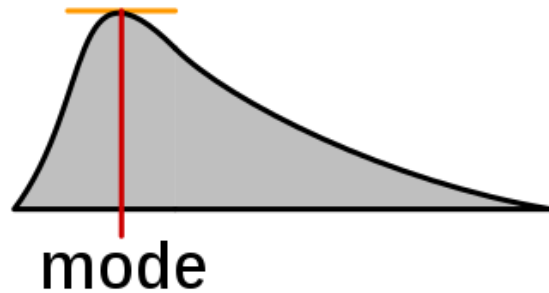
Removendo duplicatas

- Quantas linhas duplicadas existem?
 - `df.duplicated().sum()`
- Encontrar e remover linhas duplicadas:
 - `df.drop_duplicates(inplace=True)`
 - OBS.: `inplace=True` faz a alteração diretamente no próprio df, sem criar um novo DataFrame.

Tratando valores faltantes

- Quantos valores estão faltando?
 - `df.isnull().sum()`
- Técnicas para o tratamento de valores ausentes:
 - Remover linhas ou colunas
 - `df.dropna()`
 - `df.dropna(inplace=True)`
 - ✓ OBS.: altera o df original.

Tratando valores faltantes



- Técnicas para o tratamento de valores ausentes:
 - Preencher (imputar) com a média, a moda ou a mediana:
 - **Média:** É a soma de todos os valores dividida pela quantidade total de elementos.
 - ✓ Sensível à *outliers*.
 - ✓ Usada com variáveis numéricas.
 - **Mediana:** É o valor que ocupa a posição central de um conjunto de dados ordenado. Ela divide os dados em 50% acima e 50% abaixo.
 - Insensível a *outliers*.
 - Usada com variáveis numéricas.
 - **Moda:** É o valor que aparece com a maior frequência no *dataset*.
 - Insensível a *outliers*.
 - Usada com variáveis numéricas e categóricas.
 - OBS.: Técnica usada quando o número de linhas a serem removidas é muito grande em relação ao tamanho do dataset.

Tratando valores faltantes

- Preencher (imputar) com a média:
 - `num_cols = df.select_dtypes(include="number").columns`
 - Seleciona apenas as colunas numéricas do DataFrame (tipos como int64, float64, etc.).
 - Retorna uma lista com os nomes das colunas numéricas.
 - `df[num_cols] = df[num_cols].fillna(df[num_cols].mean())`
 - Pega todas as colunas numéricas e preenche os valores faltantes de cada uma com a sua própria média.

Tratando valores faltantes

- Preencher (imputar) com a mediana:
 - `num_cols = df.select_dtypes(include="number").columns`
 - Seleciona apenas as colunas numéricas do DataFrame (tipos como int64, float64, etc.).
 - Retorna uma lista com os nomes das colunas numéricas.
 - `df[num_cols] = df[num_cols].fillna(df[num_cols].median())`
 - Pega todas as colunas numéricas e preenche os valores faltantes de cada uma com a sua própria mediana.

Tratando valores faltantes

- Preencher (imputar) com a moda:
 - `num_cols = df.select_dtypes(include="number").columns`
 - Seleciona apenas as colunas numéricas do DataFrame (tipos como `int64`, `float64`, etc.).
 - Retorna uma lista com os nomes das colunas numéricas.
 - `df[num_cols] = df[num_cols].fillna(df[num_cols].mode().iloc[0])`
 - Pega todas as colunas numéricas e preenche os valores faltantes de cada uma com a sua própria moda.
 - OBS.1: A moda pode ter mais de um valor. Assim, em geral, usamos o primeiro valor (i.e., `iloc[0]`).
 - OBS.2: Note que a moda pode ser usada para preencher valores faltantes de variáveis numéricas ou categóricas. Portanto, para selecionar colunas categóricas, troque `include="number"` por `include="object"`.

Tratando valores faltantes

- Técnicas para o tratamento de valores ausentes:
 - Preenchimento inteligente: usa estimativas dos valores faltantes obtidas através do padrão dos dados, não um valor fixo.
 - O valor faltante é estimado a partir de outras colunas e linhas.
 - Pode-se usar as classes da biblioteca SciKit-Learn
 - `KNNImputer`: usa a media das K amostras mais “próximas” para predizer os valores faltantes.
 - `IterativeImputer`: treina um modelo de regressão para cada atributo (i.e., coluna) para predizer os valores faltantes.
 - OBS.: Funcionam apenas para dados numéricos.

Tratando valores faltantes

Exemplo de uso do KNNImputer

```
from sklearn.impute import KNNImputer
```

```
df_num = df.select_dtypes(include="number")
```

```
imputer = KNNImputer(n_neighbors=5) # número de vizinhos: 5
```

```
df_num_imputed = imputer.fit_transform(df_num)
```

OBS.1: O método 'fit_transform' treina o modelo e preenche os valores faltantes.

OBS.2: Em ML, calculamos o valor faltante **só no conjunto de treinamento** (para evitar *data leakage*). Isso é válido para qualquer pré-processamento/transformação.

Tratando valores faltantes

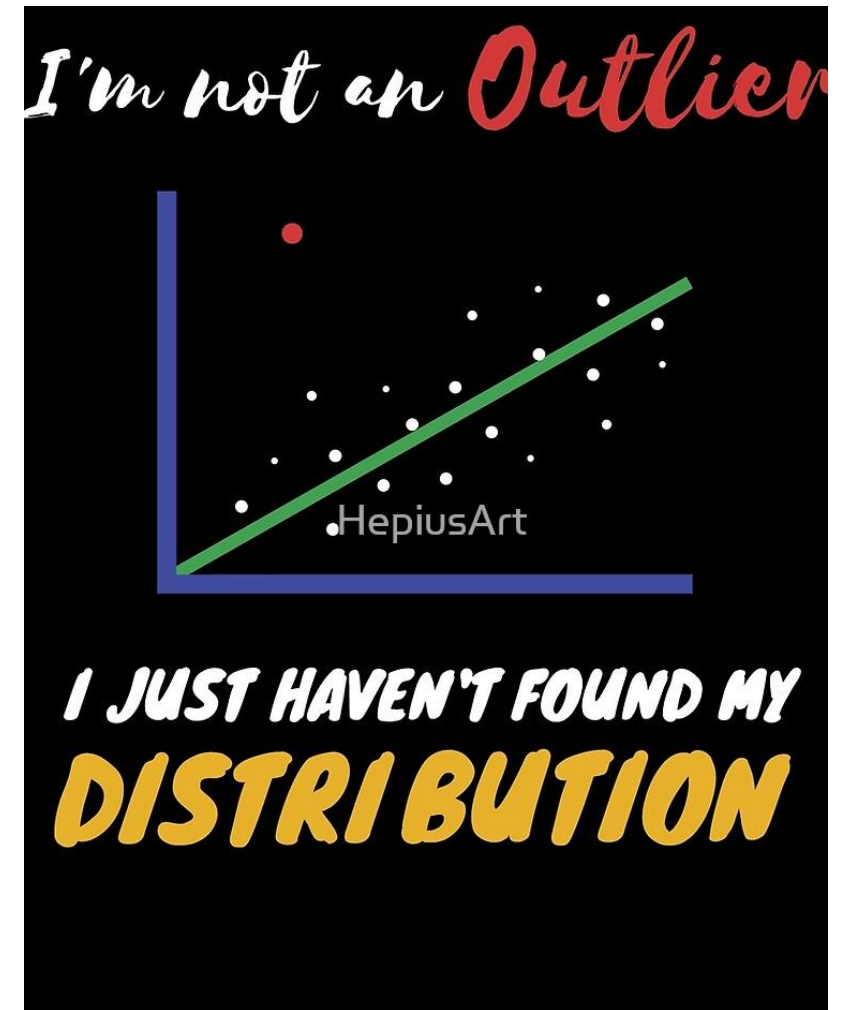
Exemplo de uso do `IterativeImputer`

```
# import necessário
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

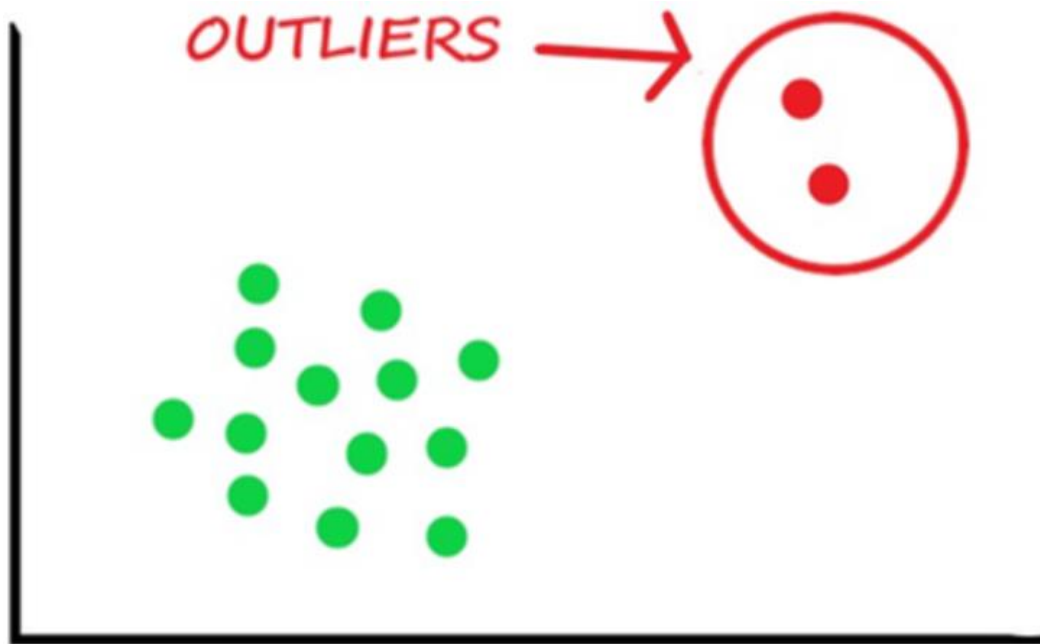
df_num = df.select_dtypes(include="number")

imputer = IterativeImputer()
df_num_imputed = imputer.fit_transform(df_num)
```

O que é um *outlier*?



O que é um *outlier*?



- Um *outlier* é uma observação que foge muito do padrão dos dados, ou seja, um valor extremamente diferente da maioria.
- Um *outlier* pode ser
 - Um erro de medição ou registro (e.g., digitação),
 - Erro de processamento dos dados (e.g., conversão errada de unidade)
 - um valor raro, mas válido (e.g., uma fraude),
 - uma cauda longa (distribuição assimétrica),
 - ou um valor de um grupo/classe diferente.

O que eles causam?

- Distorcem estatísticas
 - Afetam média, desvio padrão e correlações.
- Prejudicam visualizações
 - Esticam a escala e escondem padrões reais.
- Impactam modelos de ML, especialmente os baseados em distância
 - Geram modelos enviesados e de pior desempenho.
- Porém, podem revelar informação importante
 - Outliers podem revelar evento raro relevante como fraude, falha em sistemas, ataques cibernéticos.

O que eles causam?



r/Showerthoughts

Posted by u/ [redacted] • 2h

Bill Gates and Mark Zuckerberg are so rich the average income of Harvard dropouts is probably significantly higher than Harvard graduates.



- O post diz que a média salarial de quem largou Harvard é maior que a de quem se formou.
- Porém, largar a faculdade, provavelmente, não deixará vocês bilionários.
- O dado está correto, mas ele é uma **exceção** (não um erro) tão grande que destrói a representatividade da média.

O que eles causam?



Se estivéssemos medindo a temperatura de um servidor e um sensor marcasse 500°C por um segundo, isso é um erro ou uma exceção que deve ser investigada?

- *Outliers* são exceções.
- Porém, exceções não são necessariamente erros.
- Nunca devemos removê-los ou ignorá-los sem investigar.

Como detectar *outliers*?

- De forma visual:
 - Histograma
 - boxplot
- De forma estatística:
 - Z-score
 - IQR (intervalo interquartil)

Como detectar *outliers*?

Histograma

- Com pandas

```
df['Price'].hist(bins=30)
```

- OBS.: O parâmetro `bins` ajusta o tamanho das caixas de contagem e é opcional, por padrão, é igual a 10. Funciona bem com valores de 20 a 30.

- Com Matplotlib (mais controle)

```
plt.hist(df['Price'], bins=30, density=True)
```

```
plt.xlabel('Preço')
```

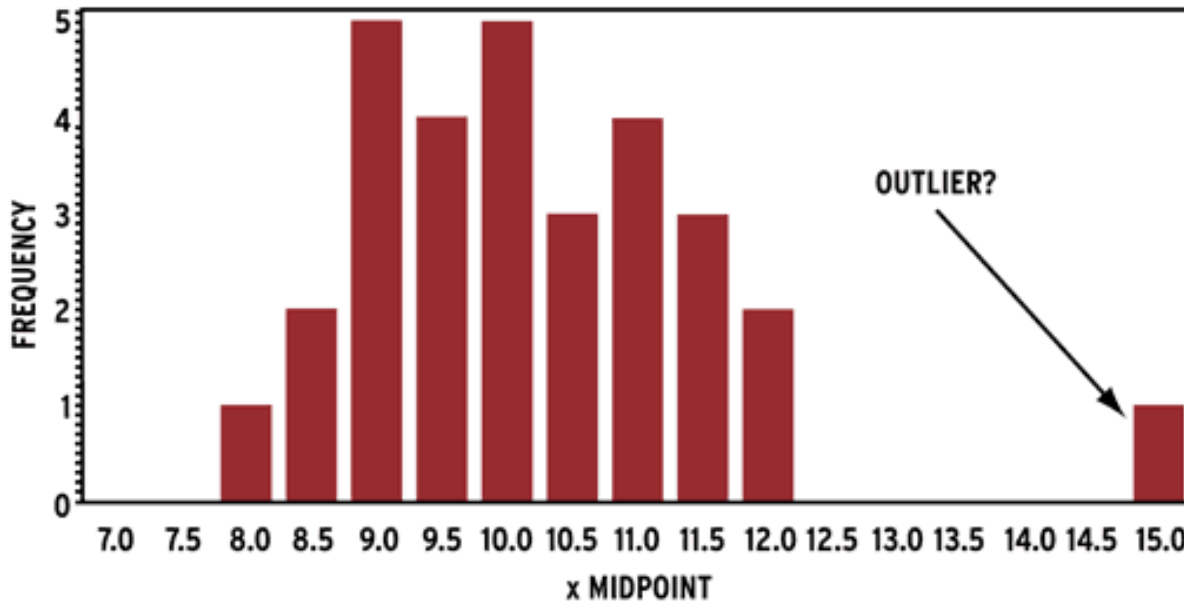
```
plt.ylabel('Frequência')
```

```
plt.title('Histograma do Preço')
```

```
plt.show()
```

- Os parâmetros `density` e `cumulative` estimam a PDF e a CDF (padrão, `False`).

Como detectar *outliers*?

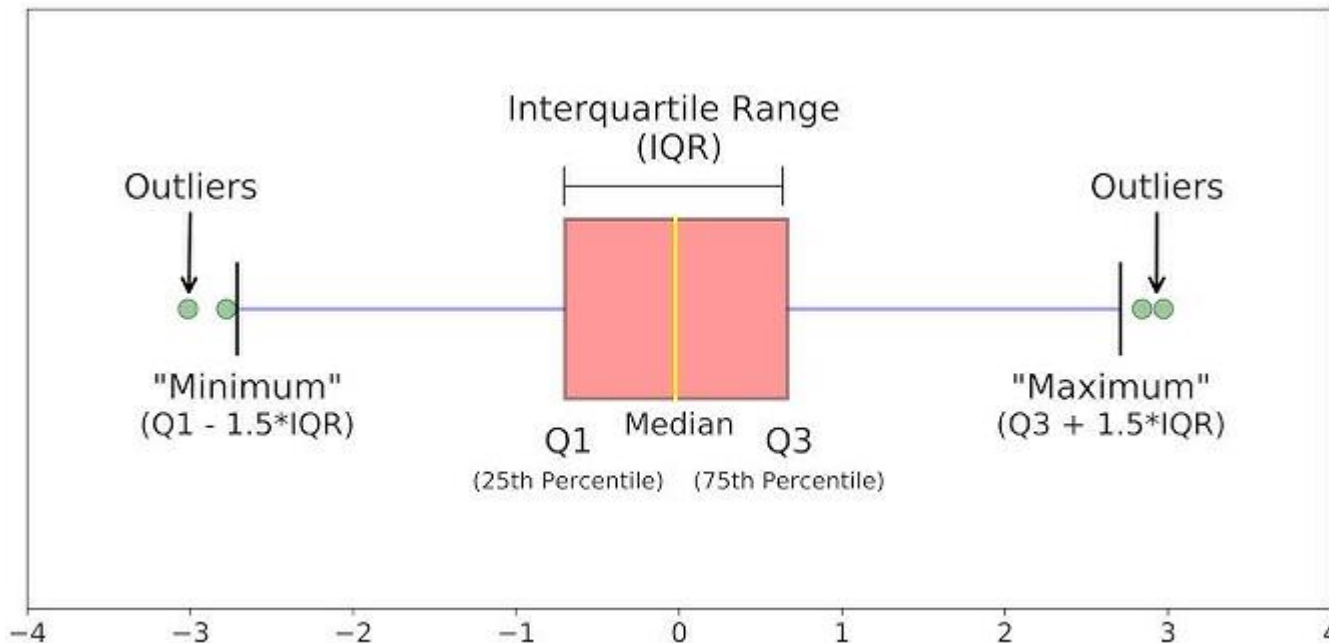


- Histogramas ajudam, mas não são precisos sozinhos.
 - Dependem do número de bins.
 - Não definem limites objetivos.
 - Um outlier pode “sumir” em um bin largo.
- Eles são ótimos para uma visão geral, sugerem a presença, mas não devem ser usados para a decisão final.
- O que fazer então?

Como detectar *outliers*?

Boxplot

Como interpretar o gráfico?

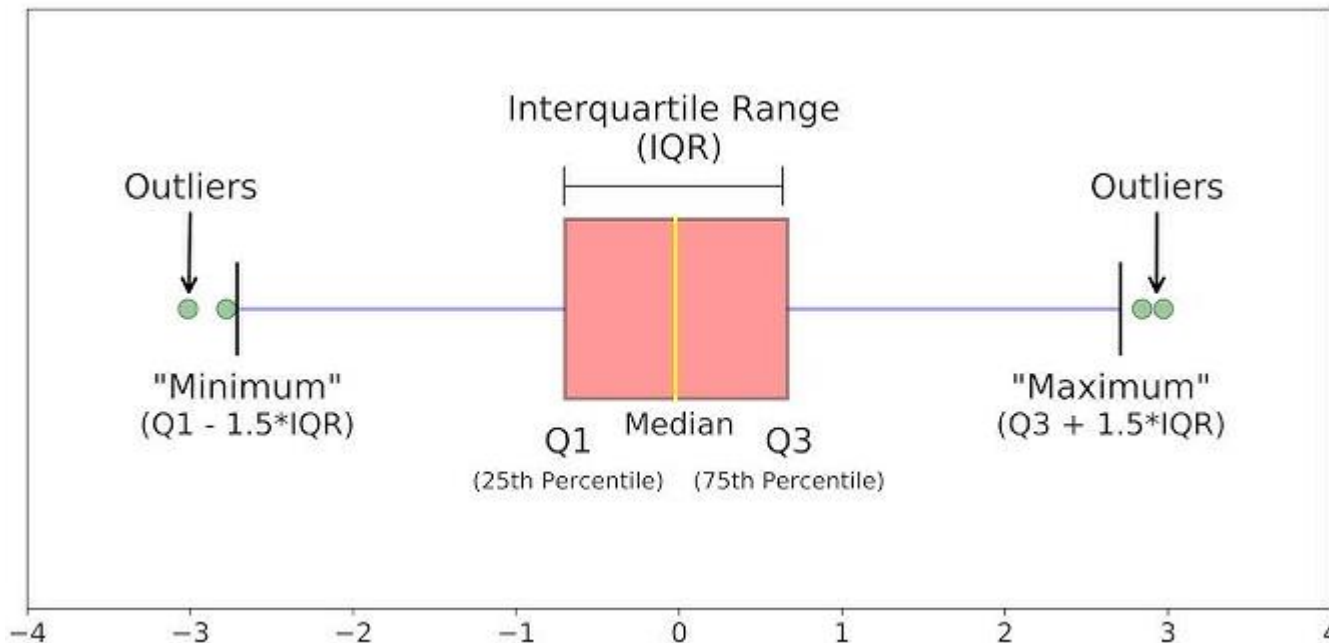


- **Borda inferior da caixa** → primeiro quartil (Q1): 25% dos dados estão abaixo desse valor.
- **Linha dentro da caixa** → mediana (Q2): valor central dos dados. Metade dos valores está acima e metade abaixo.
- **Borda superior da caixa** → terceiro quartil (Q3): 75% dos dados estão abaixo desse valor.

Como detectar *outliers*?

Boxplot

Como interpretar o gráfico?

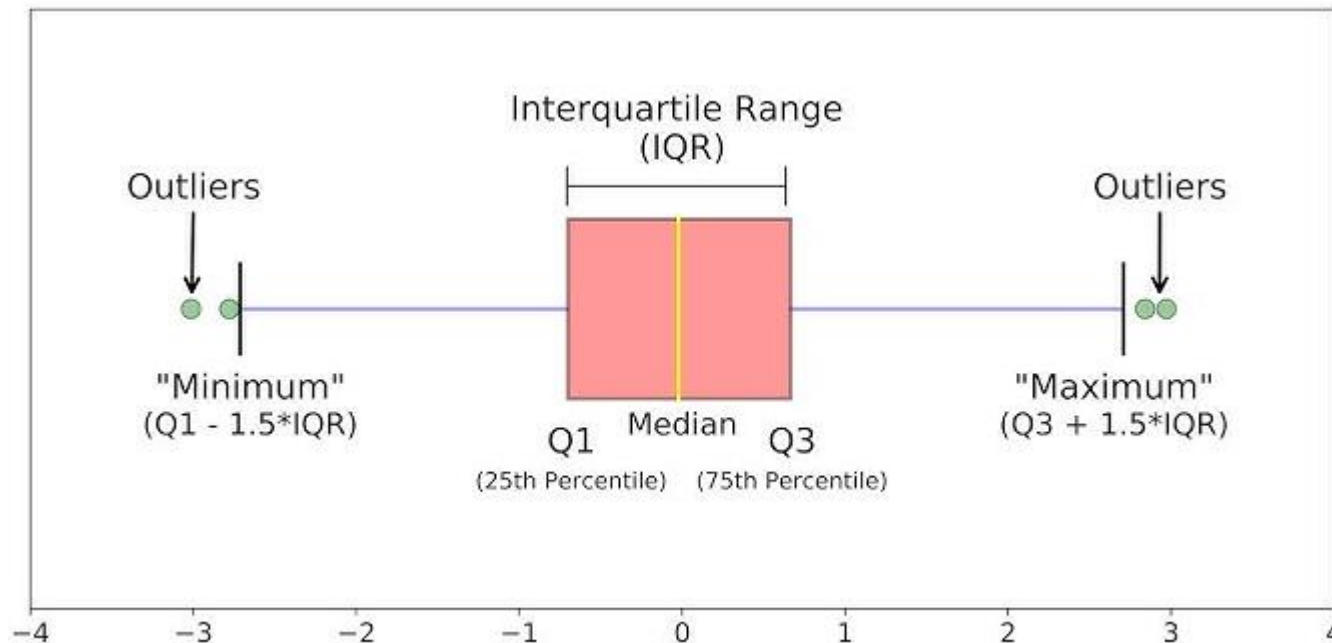


- **Caixa** → intervalo interquartil ($IQR = Q3 - Q1$): mostra a variabilidade dos dados em torno da mediana.
 - Se a mediana estiver deslocada dentro da caixa, indica **assimetria** na distribuição.

Como detectar *outliers*?

Boxplot

Como interpretar o gráfico?



- **Linhas que saem da caixa** → bigodes (*whiskers*): vão até o menor e maior valores que não são *outliers*.
 - Os limites são geralmente definidos por:
 - $Q1 - 1.5 \times IQR$
 - $Q3 + 1.5 \times IQR$
- **Pontos fora dos bigodes** → possíveis *outliers*: são os valores abaixo e acima dos intervalos anteriores.

Como detectar *outliers*?

Boxplot

- Com pandas

```
df['Year'].plot.box()
```

- Com Matplotlib (mais controle)

```
plt.boxplot(df['Year'])
```

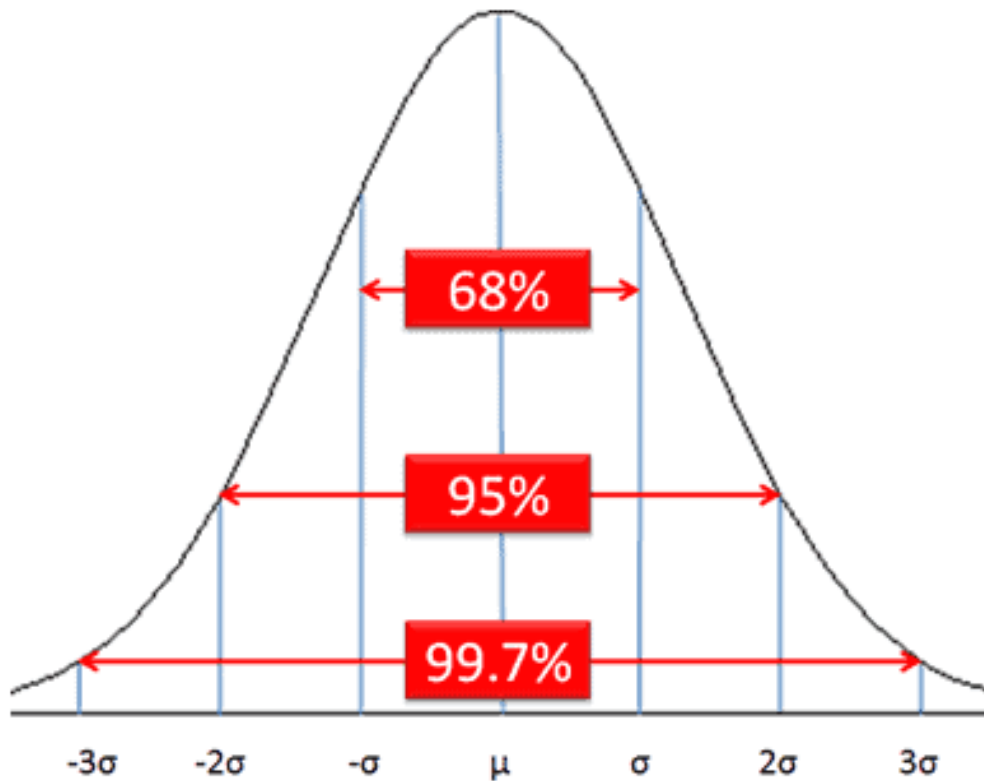
```
plt.xlabel('Year')
```

```
plt.ylabel('Valores')
```

```
plt.title('Boxplot do Year')
```

```
plt.show()
```

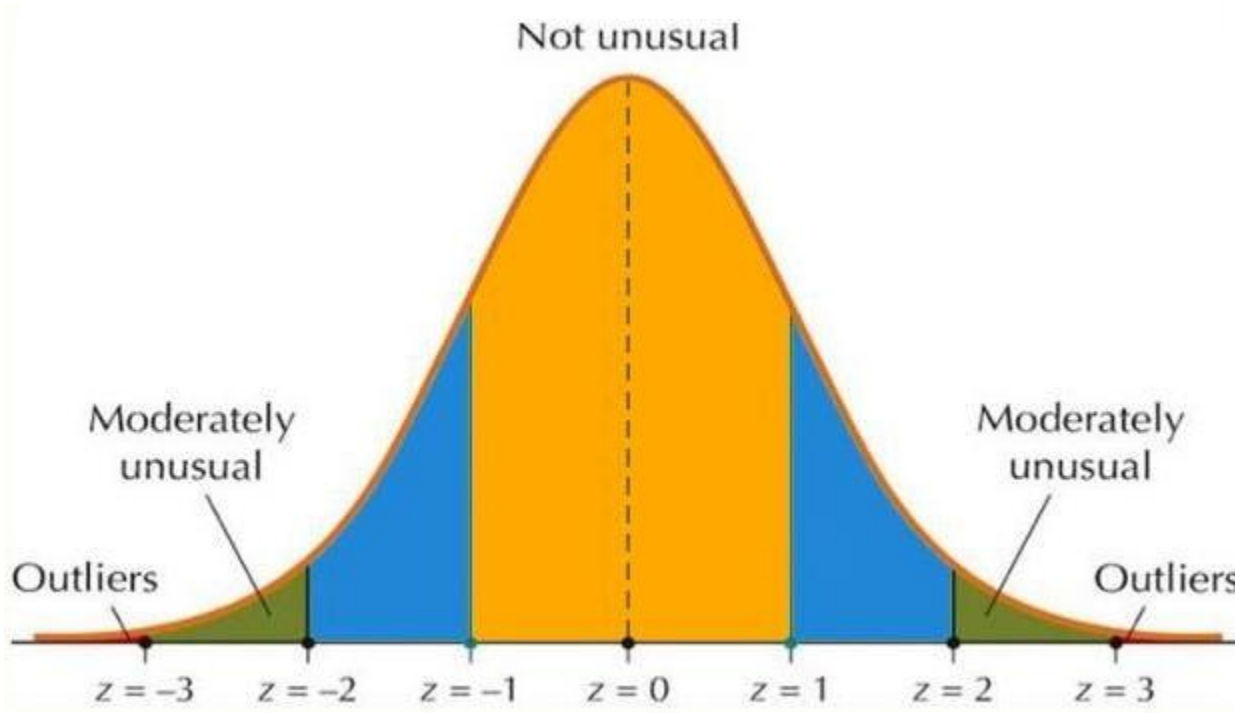
Como detectar *outliers*?



Z-score

- Mede quantos desvios-padrão um valor está distante da média da distribuição.
- Assume que os dados seguem uma distribuição semelhante à normal.
- 99.7% dos valores estão dentro do intervalo de $\pm 3\sigma$.
- Assim, assume-se que valores maiores do que $\pm 3\sigma$ são possíveis *outliers*, pois são valores raros.

Como detectar *outliers*?



Z-score

- Z-score de um valor x é calculado como:

$$z = \frac{x - \mu}{\sigma}$$

- z indica a distância até a média:
 - $z = 0 \rightarrow$ valor igual à média
 - $z = 1 \rightarrow$ 1 desvio-padrão acima da média
 - $z = 2 \rightarrow$ 2 desvios-padrão abaixo da média

Como detectar *outliers*?

Z-score

```
num_cols = df.select_dtypes(include='number').columns
z_scores = np.abs(stats.zscore(df[num_cols]))
outliers_por_linha = (z_scores > 3).any(axis=1)
df_outliers = df[outliers_por_linha]
```

Vantagens e desvantagens: boxplot (IQR)

- Vantagens

- Robusto a valores extremos, pois usa quantis e não média
- Funciona bem mesmo com distribuições assimétricas
- Fácil de interpretar visualmente (*boxplots*)
- Não assume nenhuma distribuição específica dos dados, i.e., independente da forma da distribuição

- Desvantagens

- Pode falhar em distribuições multimodais (vários picos)
- O fator 1.5 é empírico e pode não ser ideal em todos os contextos
 - Em conjuntos de dados grandes, pode definir muitos pontos como outliers ou não mostrar o verdadeiro intervalo dos dados em conjuntos menores
- Ineficaz para conjuntos de dados pequenos, pois não representam a distribuição com precisão

Vantagens e desvantagens: Z-score

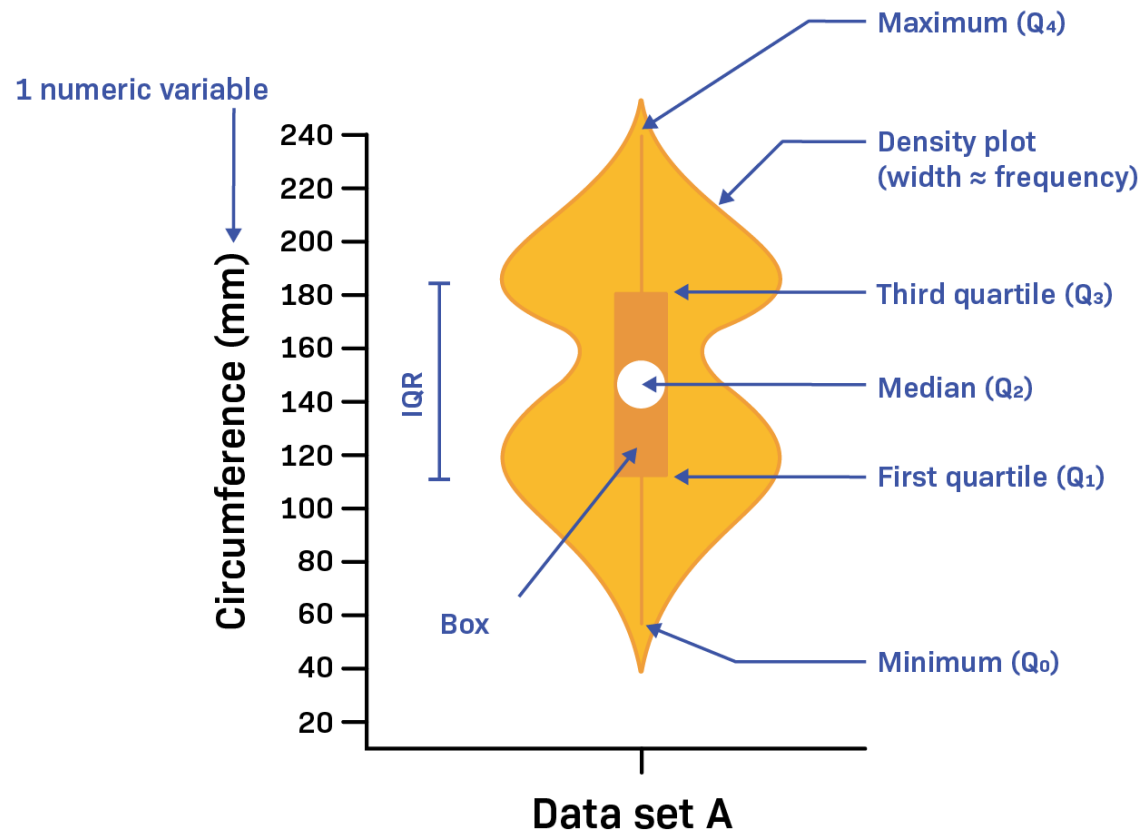
- Vantagens

- Funciona bem quando os dados seguem uma distribuição aproximadamente normal
- Permite comparar atributos em escalas diferentes (dados são padronizados)
- Simples de calcular e interpretar

- Desvantagens

- Sensível a outliers, pois média e desvio-padrão são afetados por valores extremos
- Não funciona bem com distribuições assimétricas
- Pode gerar muitos falsos positivos em distribuições assimétricas
- Parâmetros são necessários: μ e σ da população devem ser conhecidos.
 - Caso contrário, devem ser usadas estimativas da amostra, o que pode ser um problema com *datasets* pequenos.

Outros métodos para detecção de *outliers*



- Além de IQR e z-score, existem técnicas mais avançadas para detectar outliers, especialmente em dados complexos, multidimensionais ou não gaussianos.
- Métodos usando algoritmos de ML: Isolation Forest, Local Outlier Factor, One-Class SVM, Autoencoders.
- Métodos visuais: Violin plot
 - Combina *boxplot* com densidade, destacando assimetrias.

Como remover *outliers*?

- **Aplicando a distância interquartil (IQR) em várias colunas numéricas**

```
num_cols = df.select_dtypes(include='number').columns
```

```
Q1 = df[num_cols].quantile(0.25)
```

```
Q3 = df[num_cols].quantile(0.75)
```

```
IQR = Q3 - Q1
```

```
df_sem_outliers_iqr = df[  
    ~((df[num_cols] < (Q1 - 1.5 * IQR)) |  
      (df[num_cols] > (Q3 + 1.5 * IQR))).any(axis=1)  
]
```

- **OBS.:** 'any(axis=1)' retorna True se qualquer coluna daquela linha for *outlier* (i.e., True). Assim, mantemos o dataset com o mesmo número de linhas.

Como remover *outliers*?

- **Aplicando Z-score em várias colunas numéricas**

```
num_cols = df.select_dtypes(include='number').columns
```

```
z_scores = np.abs(stats.zscore(df[num_cols], nan_policy='omit'))
```

```
df_sem_outliers_z = df[(z_scores < 3).all(axis=1)]
```

- **OBS.1:** Ignora valores NaN no cálculo da média e do desvio padrão.
- **OBS.2:** `'all(axis=1)'` retorna True somente se todas as colunas daquela linha tiverem $|z| < 3$, i.e., se qualquer variável da linha for *outlier*, a linha inteira é removida.

Relação entre variáveis

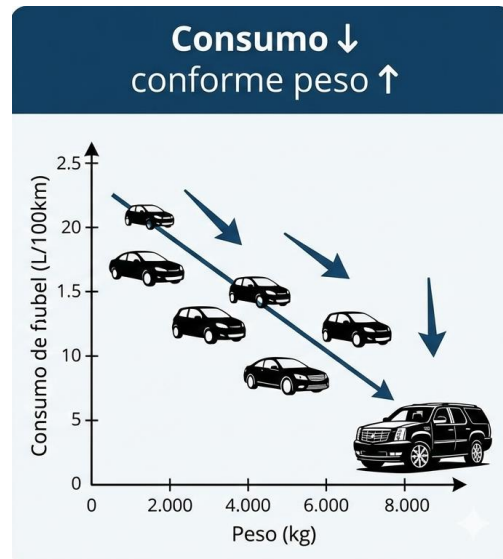
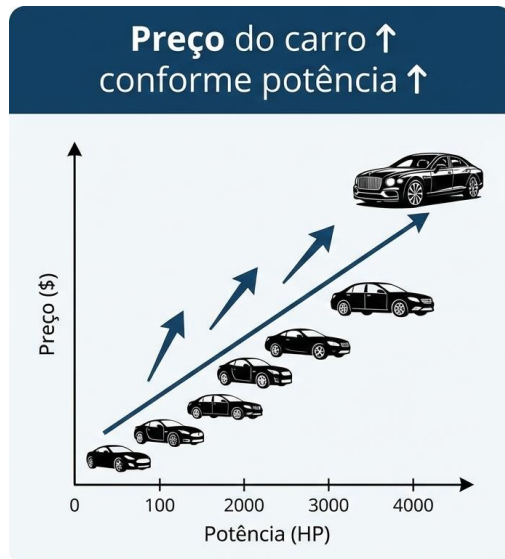


© marketoonist.com

Importante!

Correlação não implica causalidade

Relação entre variáveis

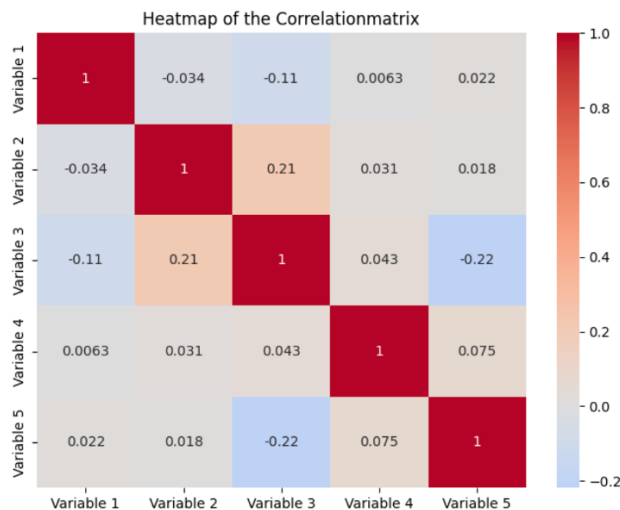
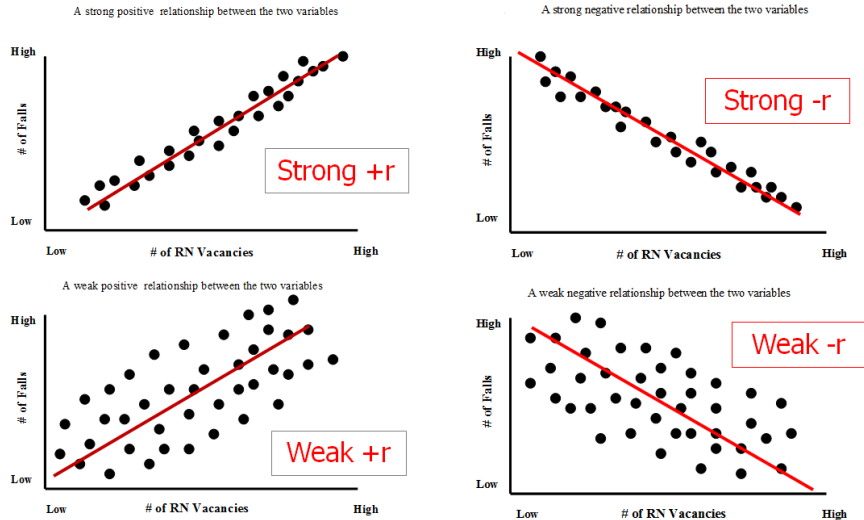


Exemplo: encontrar relações como

- Preço do carro ↑ conforme potência ↑
- Consumo ↓ conforme peso ↑

- Objetivo: descobrir padrões e estruturas escondidas nos dados.
- Relações entre variáveis revelam:
 - tendências
 - dependências
 - agrupamentos naturais
- Ao analisar relações, podemos responder:
 - O que realmente influencia o target?
 - Essa variável faz sentido nesse contexto?
 - Esse comportamento é esperado ou estranho?

Relação entre variáveis



- Relações entre variáveis ajudam a:
 - selecionar atributos relevantes
 - remover atributos redundantes
 - detectar multicolinearidade
 - Duas ou mais variáveis independentes (atributos) estão altamente correlacionadas entre si, afetando negativamente a predição dos modelos de ML.
 - escolher modelos de ML adequados
- Para analisar essas relações usamos:
 - Matriz de correlação (*heatmap*)
 - Diagrama de dispersão (*scatterplot*)

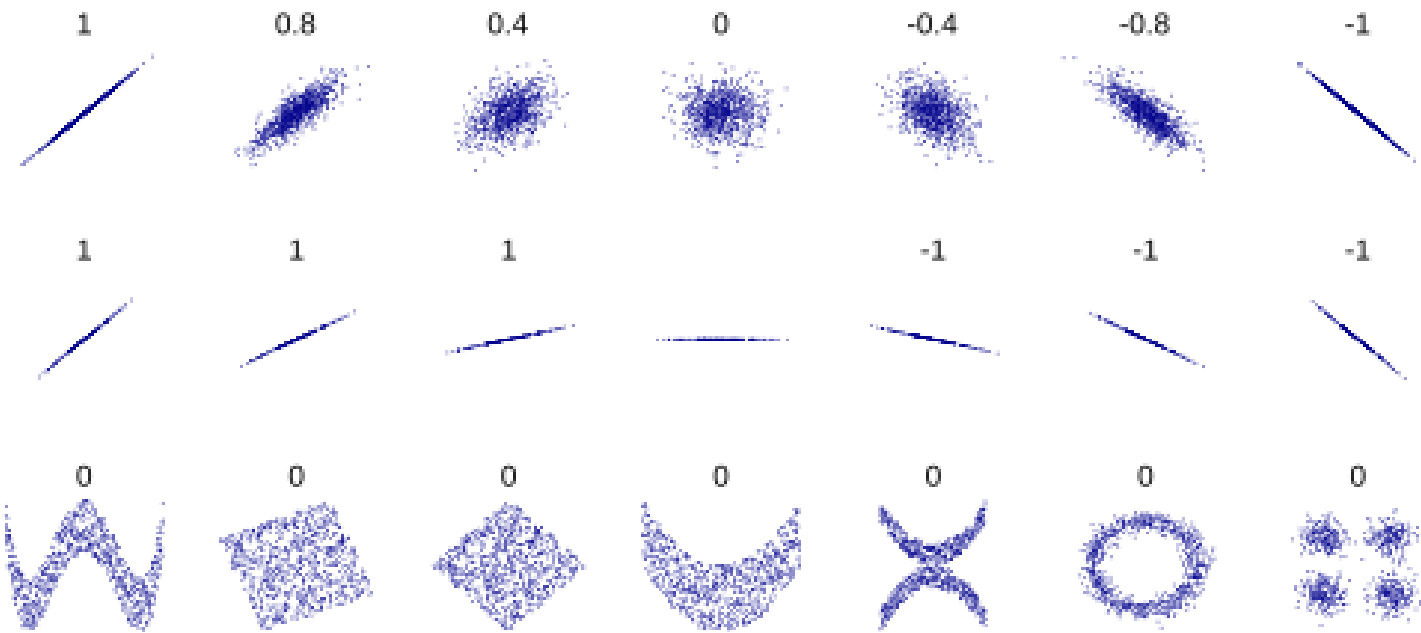
Matriz de correlação (*heatmap*)

- O *heatmap* mostra, de forma visual, o grau de **relação linear** entre pares de variáveis numéricas.
- Cada elemento da matriz é calculado usando-se o coeficiente de correlação de Pearson:

$$\rho = \frac{\sum_{i=1}^N (x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum_{i=1}^N (x_i - \mu_x)^2} \sqrt{\sum_{i=1}^N (y_i - \mu_y)^2}} \approx \frac{\text{cov}(X, Y)}{\sqrt{\text{var}(X)\text{var}(Y)}}$$

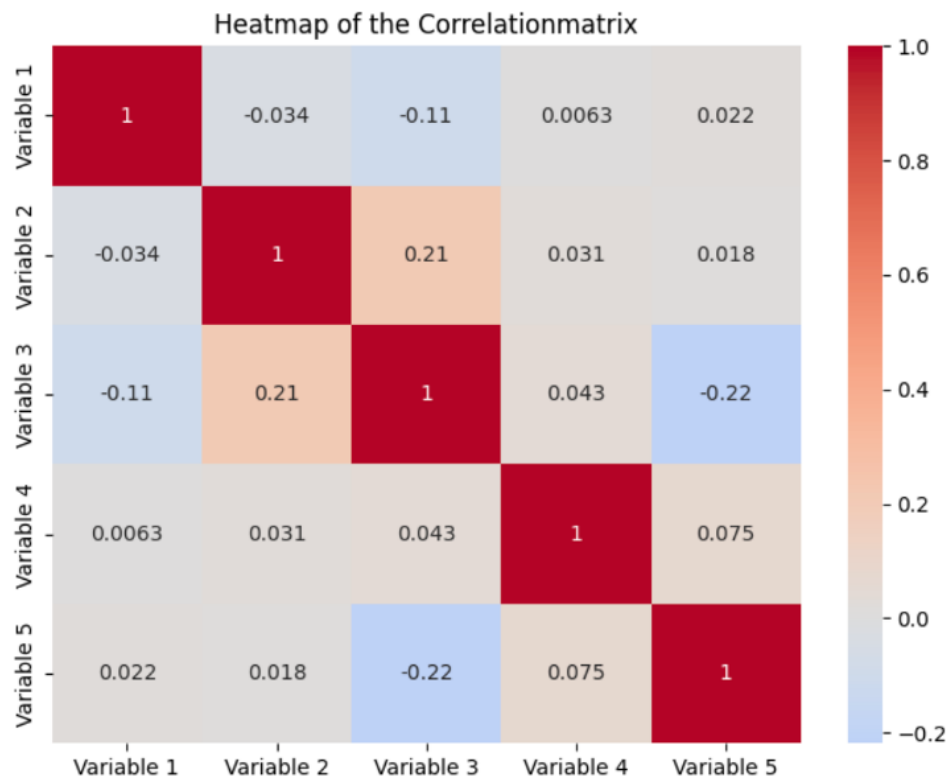
- Valores variam de -1 a +1
- **+1** → relação linear positiva forte: Y aumenta à medida que X aumenta.
- **-1** → relação linear negativa forte: Y aumenta enquanto X diminui.
- **0** → nenhuma **relação linear** entre X e Y .
- **OBS.:** Valores negativos são chamados de anti-correlação.

Correlação de Pearson



- Se ruído aumenta, então $\rho \rightarrow 0$.
 - Uma variável não fornece informação sobre a outra.
- ρ não depende da inclinação
 - Inclinações mais ou menos acentuadas resultam no mesmo ρ .
- O coeficiente de Pearson, ρ , é 0 se a correlação é não-linear
 - Mesmo y sendo uma função não linear de x (e.g., $y = \sin(x)$), ρ será 0.
- ρ é 0 se não existe relação linear.

Para que serve a matriz de correlação?



- Identificar variáveis fortemente (linearmente) relacionadas.
- Detectar multicolinearidade (↓ redundância).
- Apoiar seleção de atributos.
- Levantar hipóteses sobre relações entre variáveis.
- **Limitações**
 - ρ é sensível a *outliers* por usar a média em seu cálculo.
 - Não mostra a forma da relação e, portanto, não indica se a relação é não-linear.
 - Devemos analisar visualmente (*scatter plot*) e/ou com outro coeficiente (Spearman).

Correlação de Spearman

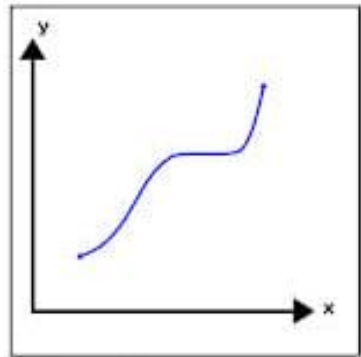
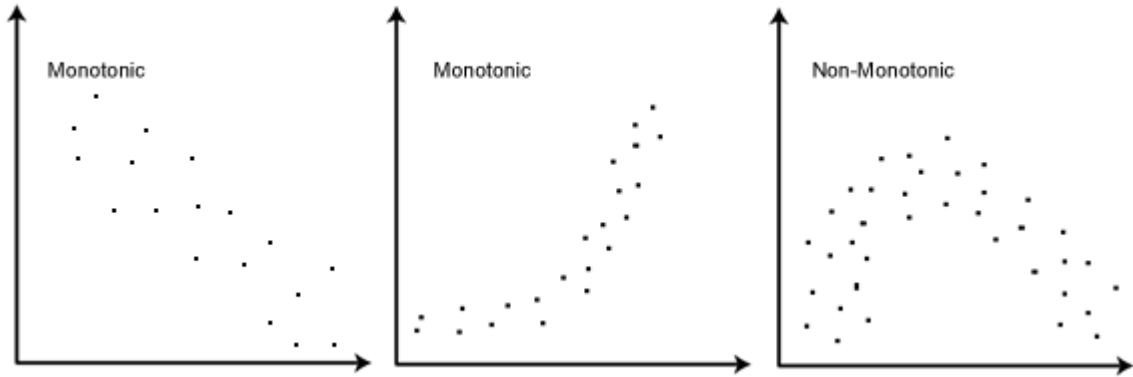


Figure 1 - A Monotonically Increasing function

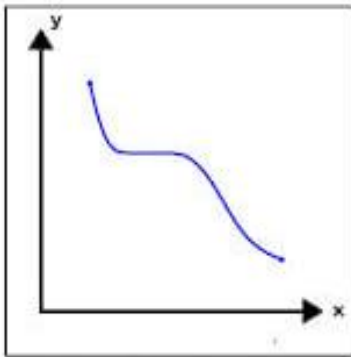


Figure 2 - A Monotonically decreasing function

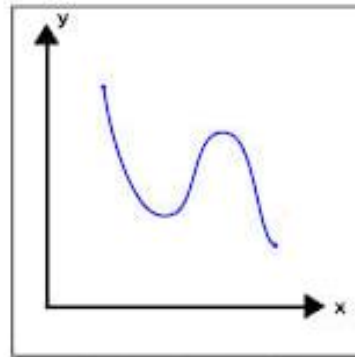


Figure 3 - A function that is not Monotonic

Se os dados crescem em formato de curva (exponencial, logarítmica), o Pearson dará um valor baixo, mas o Spearman dará 1.0.

- Mede a relação **monotônica** entre as variáveis.
 - Mede se quando uma variável sobe, a outra também sobe ou desce, independentemente se é em linha reta ou curva.
- Também avalia a relação de dados ordinais, i.e., dados que não são números exatos, mas têm uma ordem.
- Valor próximo de 0 se a relação for não-monotônica, i.e., se ela mudar de direção (e.g., $y = \sin(x)$).
- Menos sensível a *outliers*.

Plotando a matriz de correlação

- **Usando o coeficiente de Pearson**

```
corr = df.corr(method='pearson', numeric_only=True)
plt.figure()
sns.heatmap(corr, cmap="BrBG", annot=True)
plt.show()
```

- **Usando o coeficiente de Spearman**

```
corr = df.corr(method='spearman', numeric_only=True)
plt.figure()
sns.heatmap(corr, cmap="BrBG", annot=True)
plt.show()
```

Diagrama de dispersão (*scatter plot*)

- Mostra a relação direta entre duas variáveis, ponto a ponto, revelando padrões que números não mostram.
- Permite visualizar:
 - tendências (positiva ou negativa),
 - relações não-lineares,
 - *clusters* (i.e., grupos) e
 - *Outliers*

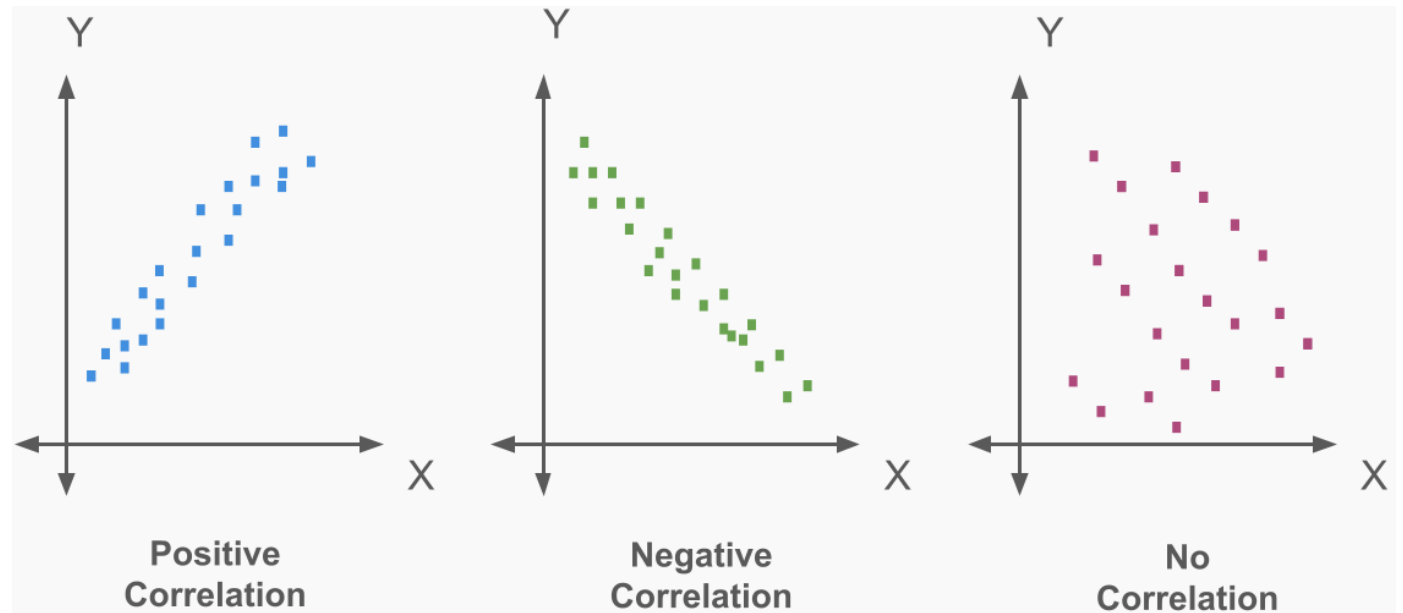


Diagrama de dispersão (*scatter plot*)

- Limitações:
 - Funciona bem para poucas variáveis
 - Difícil de escalar para alta dimensionalidade. Quando o número de variáveis cresce:
 - seriam necessários muitos scatter plots
 - a análise vira um processo manual, cansativo e propenso ao erro
 - Exemplo: 20 variáveis se tornam 190 pares possíveis
 - Pode ficar poluído em *datasets* grandes. Em datasets com muitos pontos:
 - os pontos se sobrepõem
 - regiões densas viram “manchas”
 - outliers podem ficar escondidos

Plotando o diagrama de dispersão

- **Plotando a dispersão de um par de variáveis**

```
fig, ax = plt.subplots()
ax.scatter(df['MPG-C'], df['Price'])
ax.set_xlabel('MPG-C')
ax.set_ylabel('Price')
ax.grid(True)
plt.show()
```

- **Plotando scatter plots para todos os pares de variáveis**

```
sns.pairplot(df)
plt.show()
```

OBS.: pairplot plota uma matriz de scatter plots para todos os pares de variáveis.

Pandas profiling

- O Pandas profiling (ou *ydata-profiling*) é uma biblioteca que automatiza grande parte da análise exploratória de dados (EDA) gerando um relatório completo com uma única linha de código.
- O relatório que inclui:
 - Resumo geral dos dados: número de linhas e colunas, tipos de dados.
 - Estatísticas univariadas: média, mediana, moda, desvios-padrão, histogramas.
 - Identificação de problemas: valores faltantes, duplicatas e potenciais anomalias.
 - Correlação e interações entre variáveis numéricas e categóricas.
 - Visualizações: de distribuições, barras, densidades e matrizes de correlação.
- O relatório pode ser salvo em HTML ou em um *notebook* Jupyter.

Como gerar o relatório?

```
from ydata_profiling import ProfileReport

profile = ProfileReport(df, title="Relatório de Dados")

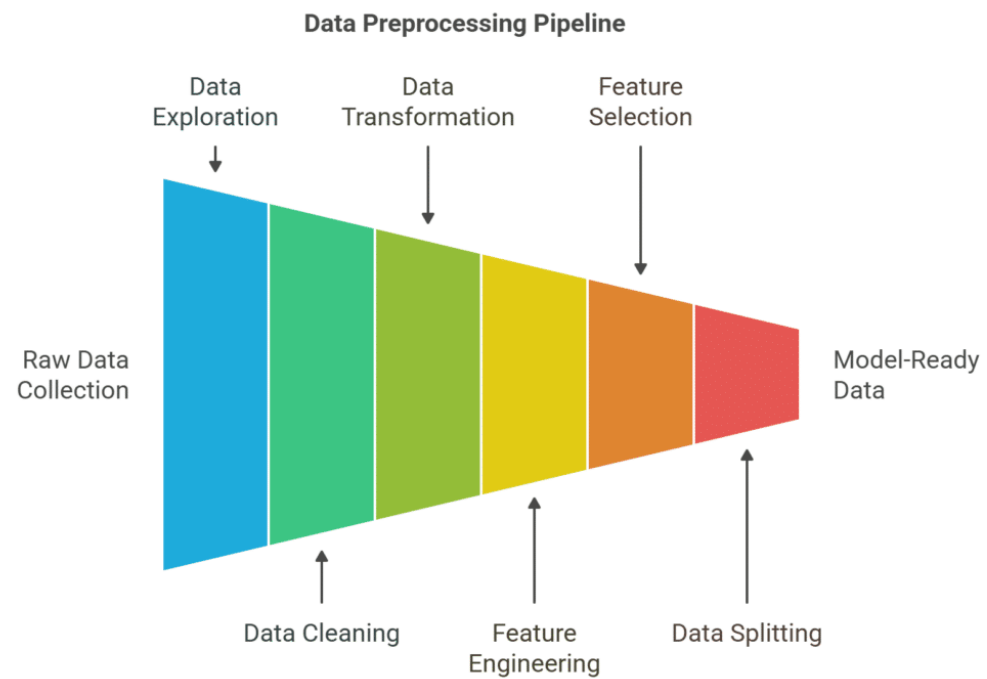
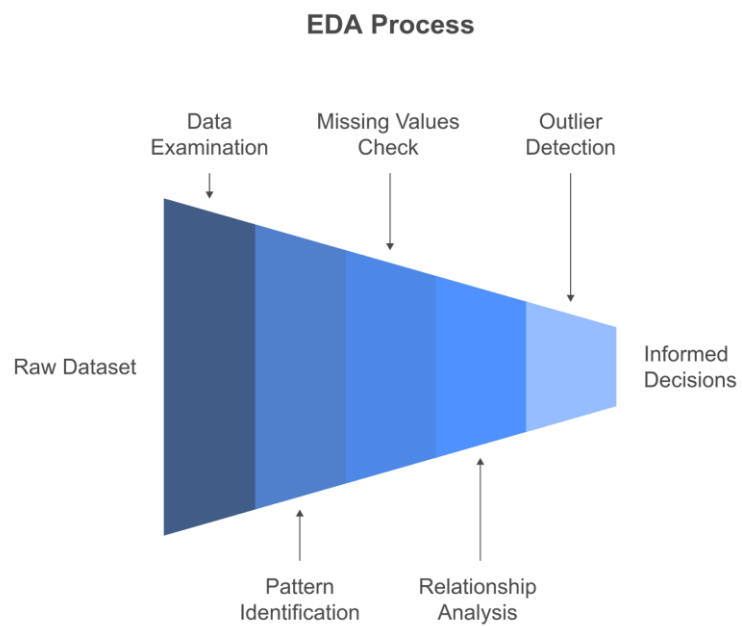
# Salva em um arquivo HTML.
profile.to_file("relatorio.html")

# Abre o relatório no notebook Jupyter.
profile.to_notebook_iframe()
```

Exemplo

- [Exemplo: intro_eda.ipynb](#)





Perguntas?

Obrigado!