

MÓDULO I

Análise exploratória com Python

Alexandre Loureiros Rodrigues

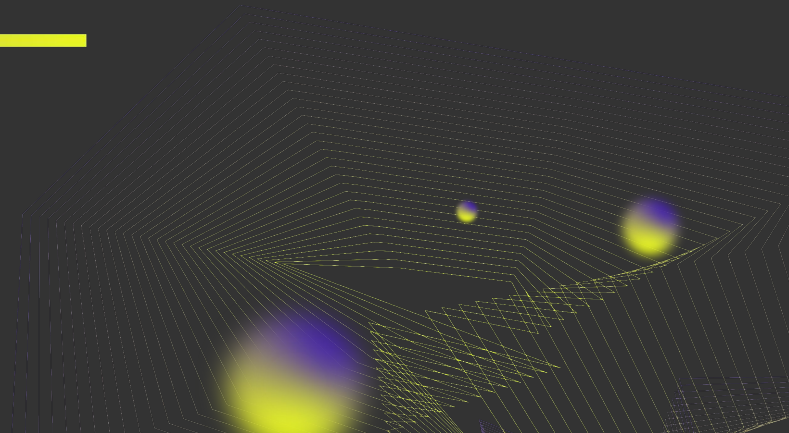
Alexandre Loureiros Rodrigues / Prof. Estatística

ESPECIALIZAÇÃO

INTELIGÊNCIA ARTIFICIAL
& CIÊNCIA DE DADOS

SEAD
UFES

Superintendência de
Educação a Distância



ÍNDICE



1. Introdução
2. Numpy
3. Pandas
4. Matplotlib
5. Scipy
6. Análise exploratória de dados

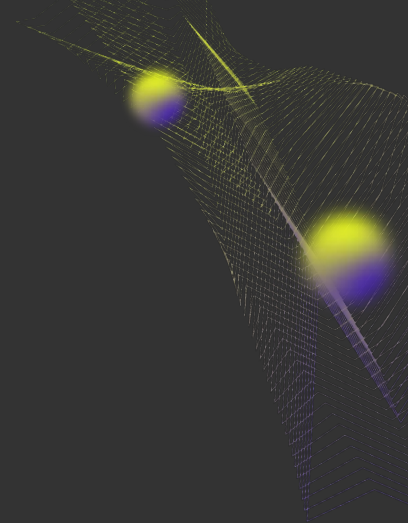


1. Introdução



Entender o conjunto de dados

- Selecionar variáveis relevantes e descartar aquelas que não agregam valor.
- Identificar outliers, dados ausentes ou erros humanos.
- Analisar as relações, ou a ausência delas, entre as variáveis.
- Ampliar os insights sobre o conjunto de dados e reduzir ao máximo possíveis erros ao longo do processo.





2. Introdução ao Numpy

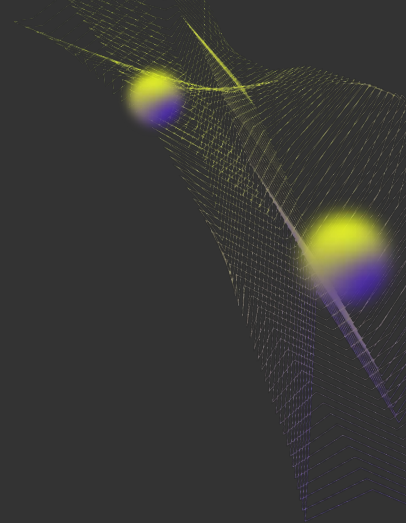


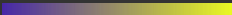
Numpy

- Biblioteca fundamental para computação científica em Python.
- Trabalha com arrays multidimensionais e fornece uma ampla gama de funções matemáticas.

Principais Funcionalidades:

- Criação e manipulação de arrays.
- Operações matemáticas eficientes.
- Ferramentas para álgebra linear, transformadas de Fourier entre outras





```
import numpy as np
```

```
# Criação de um array 3x3 de números inteiros
```

```
array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

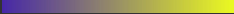
```
# Operações matemáticas
```

```
array_soma = array + 10
```

```
array_produto = array * 2
```

```
# Álgebra linear: Determinante da matriz
```

```
determinante = np.linalg.det(array)
```



Slicing e Indexing: Acesso a subarrays e elementos específicos.

Broadcasting: Aplicação de operações em arrays de diferentes dimensões.

Funções Estatísticas: Média, mediana, desvio padrão, etc.

```
# Slicing: Extraíndo subarrays
```

```
subarray = array[1:, 1:]
```

```
# Broadcasting: Operações entre arrays de diferentes formas
```

```
array_broadcast = array + np.array([1, 2, 3])
```

```
# Estatísticas: Calculando a média e o desvio padrão
```

```
media = np.mean(array)
```

```
desvio_padrao = np.std(array)
```



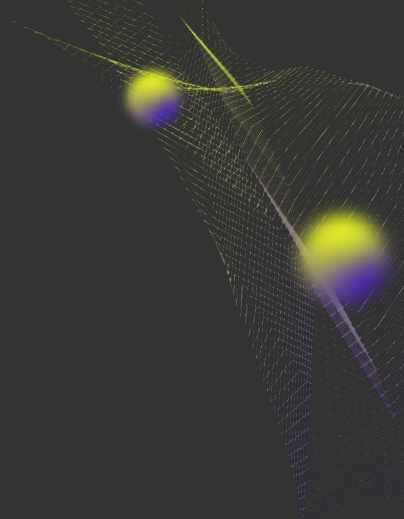

3. Introdução ao Pandas



Pandas

- Biblioteca poderosa para análise e manipulação de dados.
- Trabalha principalmente com duas estruturas de dados: Series e DataFrames.

Principais Funcionalidades:

- Leitura e escrita de dados (CSV, Excel, SQL, etc.).
 - Limpeza e transformação de dados.
 - Agrupamento e agregação de dados.
- 

```
import pandas as pd
```

```
# Criação de um DataFrame
```

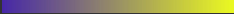
```
df = pd.DataFrame({  
    'Produto': ['A', 'B', 'C'],  
    'Vendas': [100, 150, 200],  
    'Lucro': [50, 70, 90]  
})
```

```
# Leitura de dados de um arquivo CSV
```

```
df_csv = pd.read_csv('dados.csv')
```

```
# Agrupamento de dados por produto e soma das vendas
```

```
df_agrupado = df.groupby('Produto').sum()
```



Filtros: Seleção de dados com base em condições.

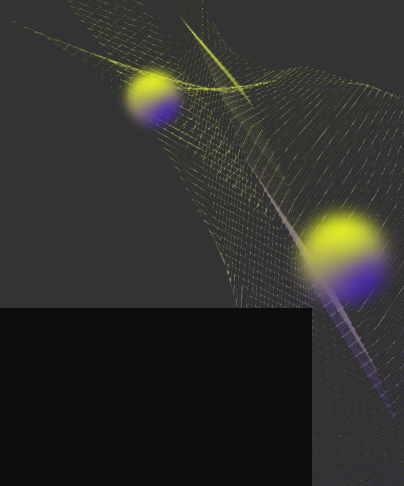
Merging e Joining:
Combinação de múltiplos DataFrames.

Trabalhando com valores nulos: Identificação, remoção ou substituição.

```
# Filtrando dados: Produtos com vendas acima de 120
df_filtro = df[df['Vendas'] > 120]

# Merge de DataFrames
df_outra = pd.DataFrame({
    'Produto': ['A', 'B'],
    'Desconto': [5, 10]
})
df_merged = pd.merge(df, df_outra, on='Produto')

# Tratamento de valores nulos: Preenchendo valores ausentes com a média
df['Lucro'] = df['Lucro'].fillna(df['Lucro'].mean())
```





4. Introdução ao Matplotlib

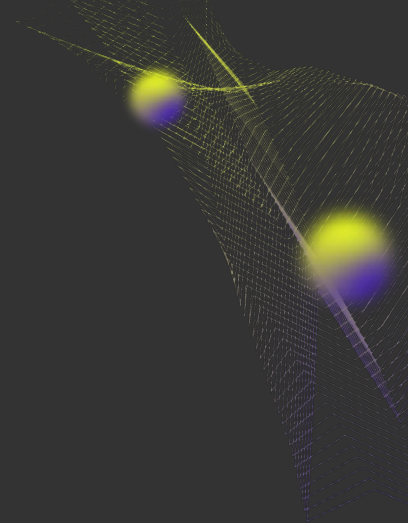


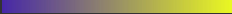
Matplotlib

- Biblioteca para criação de gráficos e visualização de dados.

Principais Funcionalidades:

- Criação de gráficos básicos (linha, barra, dispersão).
- Personalização de gráficos (cores, rótulos, títulos).
- Subplots e layouts de múltiplos gráficos.



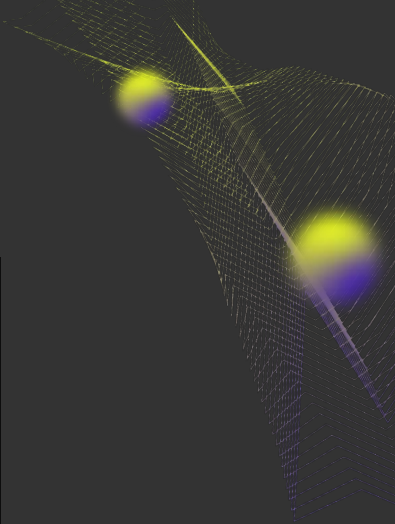


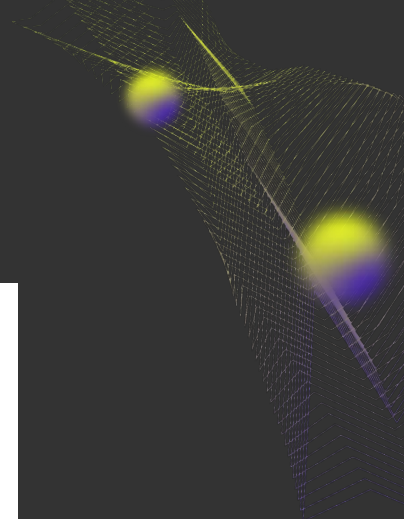
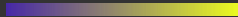
```
# Subplots: Gráfico de linha e gráfico de barras lado a lado
fig, axs = plt.subplots(1, 2, figsize=(10, 4))

# Gráfico de linha
axs[0].plot(df['Produto'], df['Vendas'])
axs[0].set_title('Vendas')

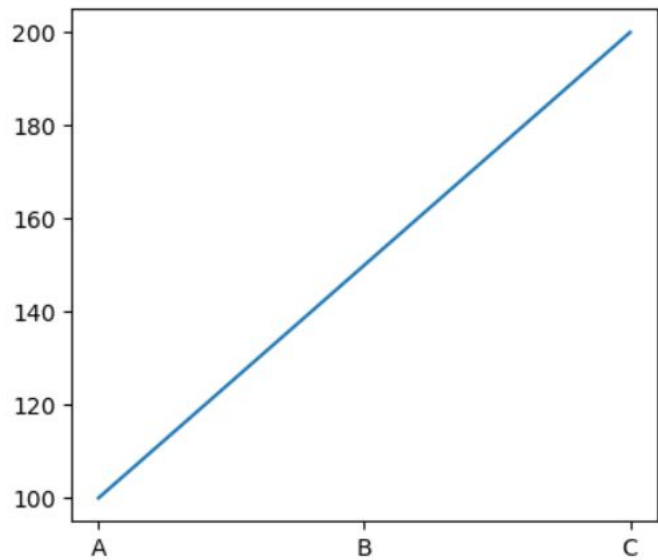
# Gráfico de barras
axs[1].bar(df['Produto'], df['Lucro'], color='orange')
axs[1].set_title('Lucro')

plt.show()
```

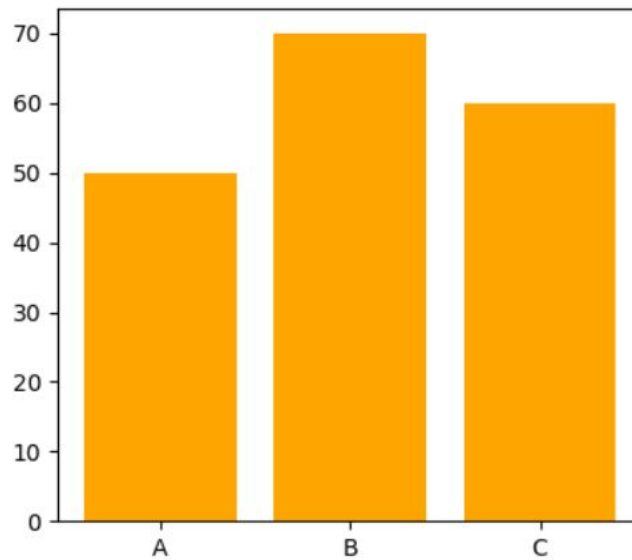




Vendas



Lucro





5. Introdução ao Scipy

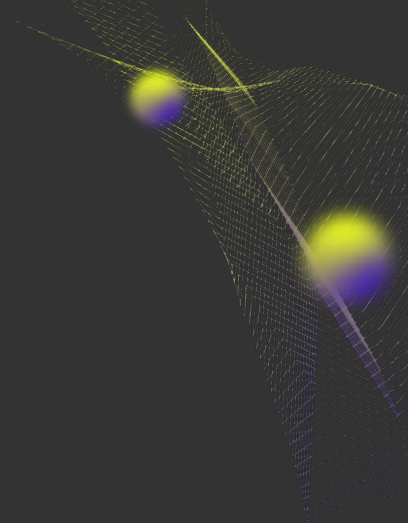


Scipy

- Biblioteca para computação científica avançada, incluindo otimização, integração, interpolação, álgebra linear, estatísticas e muito mais.

Principais Funcionalidades:

- Resolução de problemas de otimização.
- Integração e resolução de equações diferenciais.
- Funções estatísticas e geração de distribuições.





Mínimo de função quadrática

Resultado da otimização:
[-2.50000002]

```
from scipy import optimize

# Função para minimizar
def funcao(x):
    return x**2 + 5*x + 6

# Otimização (encontrando o mínimo)
resultado = optimize.minimize(funcao, 0)
print(f'Resultado da otimização: {resultado.x}')
```

Integração Numérica:

Calcular a integral de funções complexas.

Interpolação: Estimar valores entre pontos de dados conhecidos.

Análise Estatística Avançada:

Testes estatísticos, ajustes de distribuições, etc.

```
from scipy import integrate

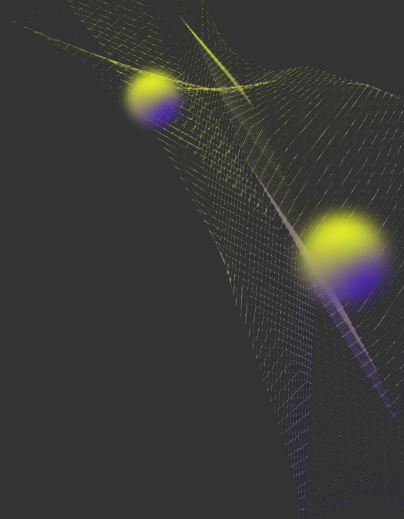
# Integração numérica (área sob a curva)
resultado_integral = integrate.quad(lambda x: x**2, 0, 5)
print(f'Resultado da integral: {resultado_integral[0]}')

# Interpolação de dados
from scipy import interpolate
x = np.linspace(0, 10, 10)
y = np.sin(x)
f_interpolacao = interpolate.interp1d(x, y)
y_interpolado = f_interpolacao(5)
print(f'Valor interpolado: {y_interpolado}')
```





Resumo

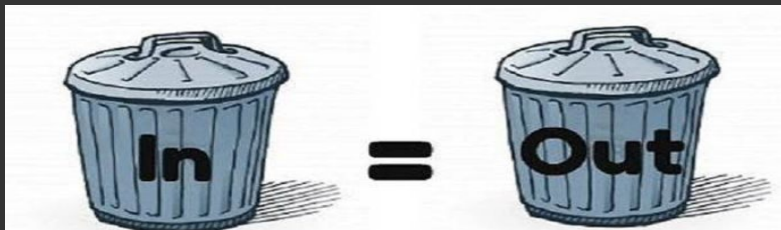
- **Numpy:** Manipulação eficiente de arrays e operações matemáticas.
 - **Pandas:** Análise e manipulação de dados estruturados.
 - **Matplotlib:** Visualização de dados.
 - **Scipy:** Computação científica avançada.
- 



6. Análise exploratória de dados

Análise exploratória para ML

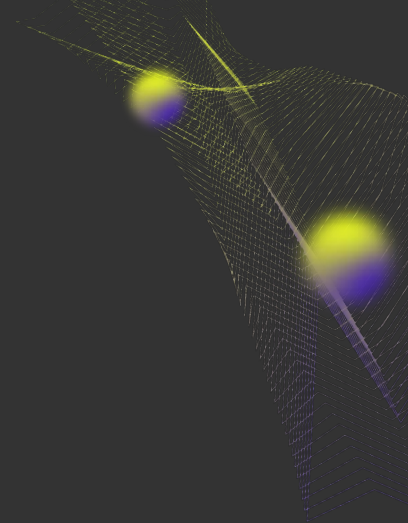
Para aplicar machine learning, é essencial avaliar a qualidade dos seus dados, identificando possíveis valores anômalos, determinando os tipos de dados disponíveis, analisando a correlação das variáveis com o target e verificando se é necessário realizar feature engineering.





Objetivos da exploração de dados

- Compreender suas variáveis e saúde dos dados (análise univariada, detecção de valores faltantes, outliers, etc)
- Limpar seu conjunto de dados caso necessário (remover variáveis desnecessárias, imputação de valores, etc)
- Analisar relações entre as variáveis (análise bivariada)





Plano de EDA para ML

1. Carregamento dos Dados

- Importar as bibliotecas necessárias.
- Carregar o conjunto de dados (CSV, Excel, etc.).
- Visualizar as primeiras linhas para entender a estrutura dos dados.

2. Análise Inicial

- Verificar o tipo de dados de cada coluna.
- Identificar valores ausentes.
- Resumir estatísticas descritivas (média, mediana, desvio padrão, etc.).



Plano de EDA para ML

1. Carregamento dos Dados

```
import pandas as pd  
df = pd.read_csv('dataset.csv') # Para CSV  
df = pd.read_excel('dataset.xlsx') # Para Excel  
  
df.head() # Exibe as primeiras 5 linhas do DataFrame  
df.tail() # Exibe as últimas 5 linhas do DataFrame
```



Plano de EDA para ML

2. Análise inicial

```
df.info() # Mostra informações sobre os tipos de dados e valores nulos  
  
df.isnull().sum() # Soma dos valores ausentes por coluna  
  
df.describe() # Estatísticas descritivas para colunas numéricas  
df.describe(include='all') # Estatísticas descritivas para todas as colunas
```



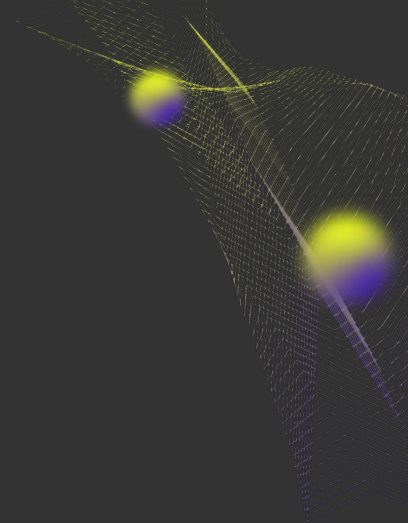
Plano de EDA para ML

3. Limpeza de Dados

- Remover dados duplicados.
- Tratar valores ausentes (remoção ou imputação).

4. Detecção e Tratamento de Outliers

- Usar métodos como Boxplot e Z-Score para detectar outliers.
- Decidir como tratar os outliers (remoção, transformação, etc.).





Plano de EDA para ML

3. Limpeza de Dados

```
df.drop_duplicates(inplace=True) # Remove duplicatas
```

```
df['column_name'].fillna(value, inplace=True) # Preenche valores ausentes  
df.dropna(inplace=True) # Remove linhas com valores ausentes
```

É preciso bastante estudo para definir o que fazer com valores ausentes (pensar na melhor forma de tratamento)

Plano de EDA para ML

4. Detecção e tratamento de outliers

```
import seaborn as sns
sns.boxplot(x=df['column_name']) # Boxplot para detectar outliers

from scipy import stats
df['zscore'] = stats.zscore(df['column_name'])
outliers = df[df['zscore'].abs() > 3] # Outliers

df = df[df['column_name'] < threshold_value] # Filtra outliers
```

Outliers precisam ser removidos? Representam uma parte da população ou são claramente erros?



Plano de EDA para ML

5. Padronização de Respostas

- Uniformizar formatos de texto (ex: maiúsculas/minúsculas, remover espaços).
- Tratar categorias e valores que representam o mesmo significado de forma diferente.

6. Feature Engineering

- Criar novas variáveis que possam ser úteis para a modelagem.
- Transformações de variáveis (log, escala, discretização, normalização, padronização etc.).



Plano de EDA para ML

5. Padronização de Respostas

```
df['column_name'] = df['column_name'].str.upper() # Converte para maiúsculas  
df['column_name'] = df['column_name'].replace({'val1': 'VAL1'}) # Corrige valores
```


Plano de EDA para ML

6. Feature Engineering

```
df['new_column'] = df['column1'] + df['column2'] # Combina colunas para criar nova
df['log_column'] = np.log(df['column_name'] + 1) # Log-transformação para suavizar
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df[['col1', 'col2']] = scaler.fit_transform(df[['col1', 'col2']])
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[['col1', 'col2']] = scaler.fit_transform(df[['col1', 'col2']])
```

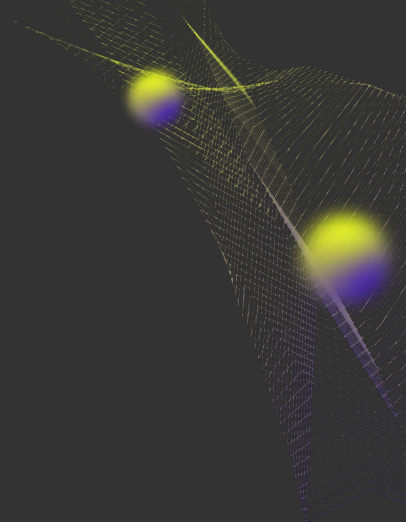
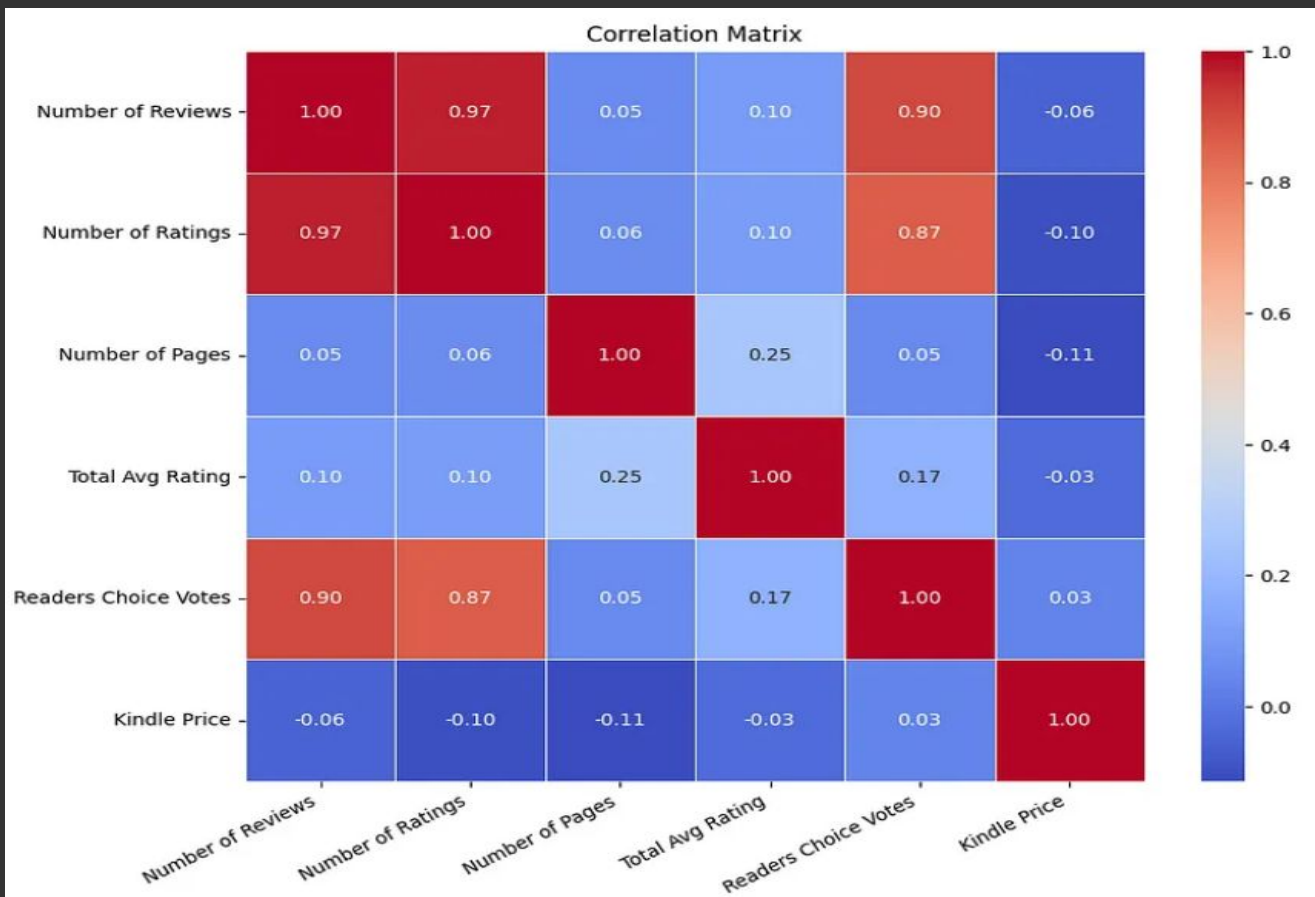


Plano de EDA para ML

7. Visualização

- Criar gráficos para melhor compreensão dos dados (histogramas, scatter plots, heatmaps, etc.).
- Analisar correlações entre as variáveis.

```
df['column_name'].hist(bins=30) # Histograma para distribuição de dados  
  
sns.scatterplot(x='column1', y='column2', data=df) # Scatter plot  
  
sns.heatmap(df.corr(), annot=True) # Mapa de calor para correlações
```



INTELIGÊNCIA
ARTIFICIAL &
CIÊNCIA DE DADOS

Alexandre Loureiros Rodrigues
Estatística

alexandre.rodrigues@ufes.br