Guia de Estudos em Ciência de Dados

> **Objetivo:** servir como material de apoio para iniciantes, cobrindo fundamentos de ciência de dados, Git e gestão de pacotes Python. O conteúdo foi organizado em ordem cronológica com base em duas aulas práticas e aprimorado para maior clareza.

Sumário

Estrutura do Fluxo de Trabalho

1.1 Ciclo Macro (CRISP-DM simplificado)

1. **Entender o negócio**
2. **Entender os dados**
3. **Preparar os dados**
4. **Modelar**
5. **Avaliar & explicar**
6. **Implantar & monitorar**

1.2 Tipos de Problema

Categoria	Subtipos	Exemplo
Supervisionado	*Classificação binária*, *multi-classe*, *Regressão*	Prever se um cliente vai atrasar (binária) ou qual espécie de flor (multi-classe)
Não-supervisionado	*Clusterização*, *Detecção de anomalias*	Agrupar clientes com comportamento de compra parecido

> **Dica:** inicie sempre definindo o *target* (variável-objetivo) antes de escolher algoritmos.

Exploração de Dados (EDA)

2.1 Inspeção Inicial

```
import pandas as pd

df = pd.read_csv("dados.csv")
print(df.shape)          # linhas × colunas
print(df.dtypes)         # tipos de dados
print(df.head())         # 5 primeiras linhas
```

- `df.isna().sum()` → checa valores faltantes.
- `df.describe()` → estatísticas básicas de colunas numéricas.

2.2 Visualização Rápida

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.pairplot(df, hue="target")
plt.show()
```

Use pairplots, histogramas e boxplots para detectar outliers e padrões.

2.3 Correlação e Redundância

```
corr = df.corr(numeric_only=True)
plt.figure(figsize=(6,5))
ax = sns.heatmap(corr, annot=False, vmin=-1, vmax=1)
ax.set_title("Matriz de Correlação")
```

- Variáveis com |cor| ≥ 0.95 costumam ser redundantes; avalie removê-las.

Pré-processamento e Preparação

3.1 Separação Features ↔ Target

```
X = df.drop(columns=["target"])
y = df["target"]
```

3.2 Escalonamento (StandardScaler)

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

> **Nota:** o `StandardScaler` assume dados aproximadamente normais. Verifique a normalidade com `scipy.stats.shapiro` quando necessário.

3.3 Particionamento Treino/Teste

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, shuffle=True)
```

Use `shuffle=False` em séries temporais para preservar a ordem.

3.4 Semente de Reprodutibilidade

```
import numpy as np
np.random.seed(42)
```

Define resultados consistentes em execuções repetidas.

Modelagem Supervisionada

4.1 Regressão Logística (multi-classe)

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=1000, multi_class="multinomial")
model.fit(X_train, y_train)
```

4.2 Árvore de Decisão

```
from sklearn.tree import DecisionTreeClassifier, plot_tree

dt = DecisionTreeClassifier(random_state=42, max_depth=None)
dt.fit(X_train, y_train)
```

Para explicar a árvore:

```
plt.figure(figsize=(12,6))
plot_tree(dt, filled=True, feature_names=X.columns, class_names=dt.classes_)
plt.show()
```

4.3 Avaliação de Métricas

```
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

y_pred = model.predict(X_test)
print("Acurácia:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Evite confiar apenas em acurácia quando as classes estiverem desbalanceadas; use F1-Score, MCC ou balanced accuracy.

Testes Estatísticos para Seleção de Features

> Conteúdo enfatizado na **Aula 2 (20 jul 2025)**.

5.1 Diagnósticos

Teste	Verifica	Quando usar
Shapiro-Wilk	Normalidade de cada grupo	n ≤ 5 000
Levene	Homogeneidade de variâncias	Pré-requisito para ANOVA

5.2 Escolha Paramétrica vs. Não-Paramétrica

1. **Se** dados → normais *e* variâncias homogêneas → **ANOVA** (F-test)
2. **Caso contrário** → **Kruskal-Wallis** (H-test)

5.3 Função-Exemplo em Loop

```
from scipy.stats import shapiro, levene, f_oneway, kruskal

def auto_test(df, feature, target):
    groups = [df.loc[df[target]==cls, feature].dropna() for cls in
df[target].unique()]
    p_norm   = max(shapiro(g).pvalue for g in groups)
    p_var    = levene(*groups).pvalue
    if p_norm > 0.05 and p_var > 0.05:
        test, p = "ANOVA", f_oneway(*groups).pvalue
    else:
        test, p = "Kruskal-Wallis", kruskal(*groups).pvalue
    return feature, test, round(p,4)

results = [auto_test(df, col, "target") for col in X.columns]
```

Utilize o p-valor para filtrar variáveis pouco informativas.

Boas Práticas com Git

6.1 Fluxo Básico

```
# dentro do diretório do projeto
git init                  # inicia repositório local
git add .                 # adiciona todos os arquivos novos
git commit -m "Mensagem breve e objetiva"
git branch -M main        # garante ramificação principal
git remote add origin https://github.com/user/repo.git
git push -u origin main   # envia primeira vez
```

Empregue `.gitignore` para arquivos de dados grandes e `requirements.txt` para dependências.

6.2 Mensagens de Commit

- Use o **imperativo** ("add confusion-matrix plot").
- Limite a primeira linha a **50 caracteres**.

Ambientes e Instalação de Pacotes

7.1 Criando Ambiente Virtual

```
python -m venv .venv
source .venv/bin/activate      # Linux/macOS
.venv\Scripts\activate         # Windows
```

7.2 Instalando Dependências

```
pip install pandas numpy scikit-learn matplotlib seaborn
pip freeze > requirements.txt
```

Use `pip install -r requirements.txt` para replicar o ambiente.

7.3 Lidando com Warnings

```
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

Apenas suprime alertas após investigar a causa real.

Exercícios Sugeridos

1. **Wine Dataset (scikit-learn)** • Reproduza todo o fluxo (EDA → testes estatísticos → modelagem). • Compare ANOVA vs. Kruskal-Wallis para cada feature.
2. **Classificação com Decision Tree** • Ajuste `max_depth` e observe impacto em *overfitting*.
3. **Git** • Crie um repositório, faça 3 commits e abra um Pull Request (PR) para si mesmo.

Referências Úteis

- Pandas – Documentation
- scikit-learn – User Guide
- Git – Book
- Python Packaging Guide

> **Próximos Passos:** pratique os exercícios, revise conceitos de estatística descritiva e experimente métricas alternativas (MCC, AUC) em problemas desbalanceados para consolidar o aprendizado.