CURSO: <BÁSICO EM MACHINE LEARNING>

- **Atividade 02 (ATIV-02)**
  - Tipo: Somativa;
  - Tema: Algoritmos básicos de programação em linguagem python e análise de dados.
  - Conteúdo: Módulo 1 e 2.
  - Participantes: Individual.
  - Avaliação do aluno.
    - Objetivo: Avaliar desempenho do aluno sobre conhecimentos básicos de programação em linguagem python e visualização e análise de dados.
    - Nota: 0 a 3 supercrítico, 4 a 6 crítico, 5 a 7 razoável e 8 a 10 bom;
    - Critérios avaliados: Respostas com coerência, coesão e com exemplos.
  - Informações adicionais: A atividade é composta por 5 questões dissertativas sobre python e 5 questões sobre visualização e análise de dados.
  - **AO CONCLUIR A ATIVIDADE: ENVIAR APENAS O LINK DO REPOSITÓRIO GITHUB (ESPECIFICAR A BRANCH) PÚBLICO.**

1. Escreva uma função que receba uma lista de números e retorne outra lista com os números ímpares.

R:
```
def pegar_impares(lista_numeros):
    impares = []
    for numero in lista_numeros:
        if numero % 2 != 0:
            impares.append(numero)
    return impares
```

2. Escreva uma função que receba uma lista de números e retorne outra lista com os números primos presentes.

R:
```
def eh_primo(numero):
    if numero < 2:
        return False

    for divisor in range(2, numero):
        if numero % divisor == 0:
            return False
    return True

def pegar_primos(lista_numeros):
    primos = []
    for numero in lista_numeros:
        if eh_primo(numero):
            primos.append(numero)
    return primos
```

3. Escreva uma função que receba duas listas e retorne outra lista com os elementos que estão presentes em apenas uma das listas.

R:
```
def apenas_em_uma(lista1, lista2):
    resultado = []

    for item in lista1:
        if item not in lista2 and item not in resultado:
            resultado.append(item)

    for item in lista2:
        if item not in lista1 and item not in resultado:
            resultado.append(item)

    return resultado
```

4. Dada uma lista de números inteiros, escreva uma função para encontrar o segundo maior valor na lista.

R:
```
def segundo_maior(lista_numeros):
    lista_sem_repetidos = list(set(lista_numeros))
    lista_sem_repetidos.sort()

    if len(lista_sem_repetidos) < 2:
        return None

    return lista_sem_repetidos[-2]
```

5. Crie uma função que receba uma lista de tuplas, cada uma contendo o nome e a idade de uma pessoa, e retorne a lista ordenada pelo nome das pessoas em ordem alfabética.

R: def ordenar_por_nome(lista_pessoas):

return sorted (lista_pessoas, key=lambda pessoa: pessoa[0].lower())

6. Como identificar e tratar outliers em uma coluna numérica usando desvio padrão ou quartis?

R:

Outliers são observações muito distantes do padrão do conjunto e podem distorcer média, desvio padrão e modelos. Eu identifico de duas formas comuns:

Quartis (IQR): compara valores com o intervalo entre Q1 e Q3; quem ficar muito abaixo de Q1 ou muito acima de Q3 é suspeito. É bom porque funciona melhor quando os dados não são “bem comportados” (não-normal).

Desvio padrão (z-score): mede a distância do valor em relação à média; valores muitos desvios acima/abaixo são candidatos a outlier (mais usado quando a distribuição é próxima da normal).

Para tratar, primeiro verifico se é erro (digitação/coleta). Se for erro, corrijo ou removo. Se for real, posso manter, ou reduzir o impacto (ex.: limitar valores extremos ou usar mediana/transformações), dependendo do objetivo da análise.

7. Como concatenar vários DataFrames (empilhando linhas ou colunas), mesmo que tenham colunas diferentes?
   Dica: Utiliza-se pd.concat() especificando axis=0 (linhas) ou axis=1 (colunas). Quando há colunas diferentes, os valores ausentes são preenchidos com NaN.

   R: Concaternar é unir DataFrames.
   axis=0 (linhas): empilha um DataFrame embaixo do outro, útil quando são “mais registros” do mesmo tipo (ex.: meses diferentes).

   axis=1 (colunas): coloca lado a lado, útil quando são “mais atributos” do mesmo conjunto (ex.: dados de vendas + dados de clima).
   Quando os DataFrames têm colunas diferentes, o pandas mantém todas as colunas e preenche com NaN o que não existir em cada parte, indicando “dado ausente” e evitando inventar valores.

8. Utilizando pandas, como realizar a leitura de um arquivo CSV em um DataFrame e exibir as primeiras linhas?

R: Eu leio o CSV para transformar em DataFrame, porque isso permite análise e manipulação. Em seguida eu visualizo as primeiras linhas para validar rapidamente se: colunas foram lidas corretamente, separador/encoding estão certos e se os tipos de dados parecem coerentes (ex.: números não viraram texto).

9. Utilizando pandas, como selecionar uma coluna específica e filtrar linhas em um “DataFrame” com base em uma condição?

R: Selecionar uma coluna é isolar uma variável específica para análise (ex.: só “preço” ou só “idade”). Filtrar linhas é aplicar uma regra lógica para manter apenas registros que atendem uma condição (ex.: vendas > 0, idade ≥ 18). Isso é base para análises direcionadas e para preparar dados antes de gráficos e modelos.

10. Utilizando pandas, como lidar com valores ausentes (NaN) em um DataFrame?

R: **NaN** representa valor faltando (não informado, erro de coleta, dado inexistente). Primeiro eu avalio **quantidade e padrão** (se é pouco ou muito, se está concentrado em uma coluna). Depois escolho uma estratégia:

- **Remover** linhas/colunas quando a falta é pequena e não compromete a análise, ou quando a coluna está “quase toda vazia”.
- **Preencher (imputar)** quando faz sentido: média/mediana para numéricos, moda para categóricos, ou valor padrão quando é um indicador (ex.: 0 = “não ocorreu”).
- **Manter NaN** quando a ausência tem significado ou quando o tratamento pode enviesar o resultado, sempre verificando impacto nas métricas e análises.