Atividades_2024-03-08

March 8, 2024

1 Exercício 1

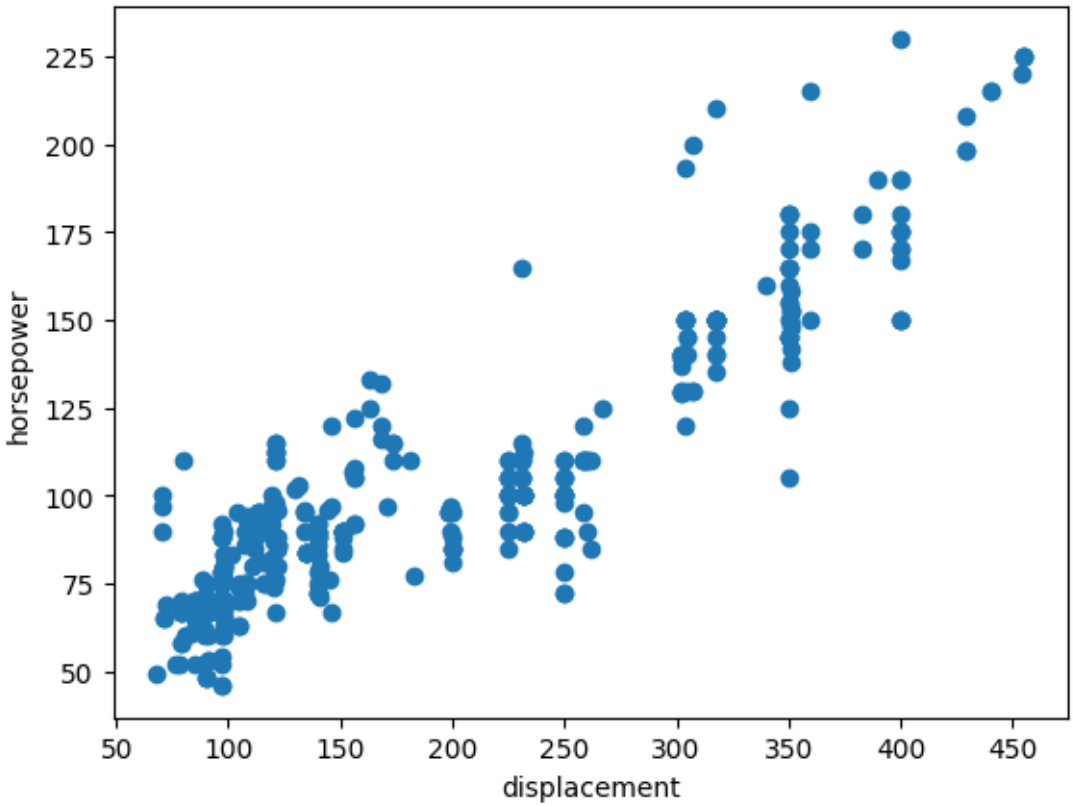
1.1 Item A

[2]:
```
import matplotlib.pyplot as pyplot
import pandas as pd
import seaborn as sea

mpg = sea.load_dataset('mpg')
diamonds = sea.load_dataset('diamonds')

pyplot.scatter(x=mpg['displacement'], y=mpg['horsepower'])
pyplot.xlabel('displacement')
pyplot.ylabel('horsepower')
pyplot.show()

mpg_filtered = mpg[mpg['cylinders'] == 8]
diamonds_filtered = diamonds[diamonds['carat'] > 3]
```



1.2 Item B

Confira as dicas sobre o uso do Jupyter Notebook em https://www.makeuseof.com/ jupyter-notebook-tips-tricks/ e compartilhe uma que tenha chamado su atençãoo

Os itens 4 (customização) e 7 (dicas para eficiência e performance) foram os elementos que gostei no texto. No caso do item 4 estou utilizando e pesquisando mais ações para customizar e tornar mais agradável o ambiente.

1.3 Item C

Sintetize com base em https://towardsdatascience.com/jupyter-notebooks-avoid-making-tos erros comuns em Jupyter Notebook

Os **erros mais comuns** ao se utilizar Jupyter Notebooks, segundo o autor do texto, são:

- **Executar células fora de ordem** A execução de células do notebook fora de ordem podem levar a exibição de erros ou exibir resultados inesperados. É uma boa prática organizarmos as células de maneira que, quando executadas, reproduzam exatamente o mesmo resultado.

- **Colocar muitas coisas em uma célula** Inserir muitas informações e saídas em uma célula faz com que a interatividade seja suprimida. Uma boa vantagem do uso da ferramenta, por exemplo, é poder ver o que ocorre em cada etapa de um script. Um bom planejamento a se ter é fazer com que cada célula conclua uma tarefa e imprima informaçõe importantes para informar se o código funcionou conforme o esperado.
- **Aproveitar o recurso de células Markdown** O recurso de Markdown adicionar possibilidades infinitas aos notebooks, como criar textos explicativos, documentação, expor ideias, entre outras possibilidades. Deixar de utilizá-los é jogar fora todas estas vantagens.
- **Importar bibliotecas no meio do código** Devemos evitar inserir declarações de importação de bibliotecas em células no méio do notebook. Isso dificulta a leitura, pode causar erros e dificultar a exportação de códigos para ambientes de produção. Além de ser uma prática não muito profissional.

1.4 Item D

Sintetize com base em https://www.linkedin.com/advice/3/what-best-ways-troubleshoot-ealgumas formas para diagnosticar os erros em Jupyter Notebook.

1. Ler as informações da exceção A primeira coisa a se fazer para investigar um problema é ler com atenção as informações da exceção.
2. Utilizar o comando %debug Este recurso habilita o modo de execução interativo, permitindo a inspeção de variáveis, executar comandos e rastrear o fluxo de execução do código.
3. Utilizar ferramentas e bibliotecas externas ao Jupyter para solução de erros Algumas vezes as ferramentas para depuração do Jupyter podem não ser suficientes para solucionar erros complexos. Exemplos de ferramentas externas: pdb (depurador Python padrão), ipdb (versão aprimorada do pdb), pytest (execução de testes automatizados).
4. Seguir boas práticas Seguir boas práticas de programação e utilização dos notebooks ajudam a depurar os códigos.

---

2 Exercício 02

[ ]:
```python
import random

tentativas = 5
continuar_jogando = 'S'
acertou = False
resultados = []

def imorime_resultados(resultados):
    for i in resultados:
        print('Número sorteado: ', i['numero_sorteado'])
        print('Tentativas: ')
```

```
        for numero in i['tentativas']:
            print(numero, end=' ')

        #Quebra de linha
        print('')

while continuar_jogando == 'S':
    numero_sorteado = random.randint(1, 100)
    estatisticas = {'numero_sorteado': numero_sorteado, 'tentativas': []}

    for i in range(1, tentativas+1):
        numero = int(input(f"Digite um número (tentativa {i} de {tentativas}:␣
 ↪"))
        estatisticas['tentativas'].append(numero)

        if numero == numero_sorteado:
            acertou = True
            break
        elif numero > numero_sorteado:
            print("O número informado é maior do que o número sorteado")
        else:
            print("O número informado é menor do que o número sorteado")

    if acertou == True:
        print("Parabéns você acertou!")
        acertou = False
    else :
        print(f"Você não acertou. O número sorteado foi {numero_sorteado}")

    resultados.append(estatisticas)
    continuar_jogando = input('Deseja continuar jogando? S - Sim ou N - Não')

#Resultados finais
print("*** Resultados finais ***")
imorime_resultados(resultados)
```