Seaborn e scatterplot

Transcrição

Queremos gerar **gráficos estatísticos** para conseguirmos entender melhor o ponto das gorjetas.

Nossa primeira análise verifica se de fato **o valor da conta influencia no valor da caixinha**. Primeiro, importaremos a nova biblioteca chamada **Seaborn**; toda feita em Python, é criada para desenhar os gráficos estatísticos que permite a exploração da distribuição de dados.

Há uma biblioteca bastante conhecida chamada **MatPlotlib**, a qual gera visualizações bastante interessantes, mas para que sejam sofisticadas, é necessário inserir uma grande quantidade de código. Porém, apesar de ser em Python, não foi criada para trabalhar com o Pandas, pois surgiu dez anos antes deste.

Em nosso projeto, utilizamos o tipo `DataFrame` como vimos anteriormente. O Seaborn surgiu neste contexto, para que possa trabalhar com `DataFrame` em um código bem mais enxuto de maior nível.

O Seaborn que importaremos é uma API feita com base no MatPlotlib para **plotagem de dados**, **geração de visualizações** e **gráficos estatísticos**, além de ser em Python.

Começaremos adicionando uma célula de texto com "# Importando o Seaborn". Para realizar a importação, utilizaremos `!pip install` seguido de `seaborn`, passando a versão `0.9.0` que

queremos com `==` .

```
!pip install seaborn==0.9.0
```

COPIAR CÓDIGO

Executaremos este comando e aguardaremos a instalação da biblioteca.

Com isso, precisaremos importar para nosso projeto adicionando `import seaborn as` com sua nomenclatura por convenção **`sns`** .

```
import seaborn as sns
```

COPIAR CÓDIGO

Criaremos outra célula textual para a primeira análise escrevendo "Análise 1 - Valor da conta e gorjeta", pois o Seaborn já está pronto para nosso projeto.

Há uma forma de abrir ou fechar todas as sessões, ajudando na organização da nossa visualização. Para isso, basta entrarmos em "*View* > *Collapse sections*" ou "*View* > *Expand sections*" na barra superior de opções.

Começaremos nossa **primeira análise** relembrando quais são as nossas colunas escrevendo `gorjeta.columns` . Geraremos um gráfico estatístico a partir dos valores das colunas `valor_da_conta` e `gorjeta` , verificando se seu aumento ou diminuição são proporcionais.

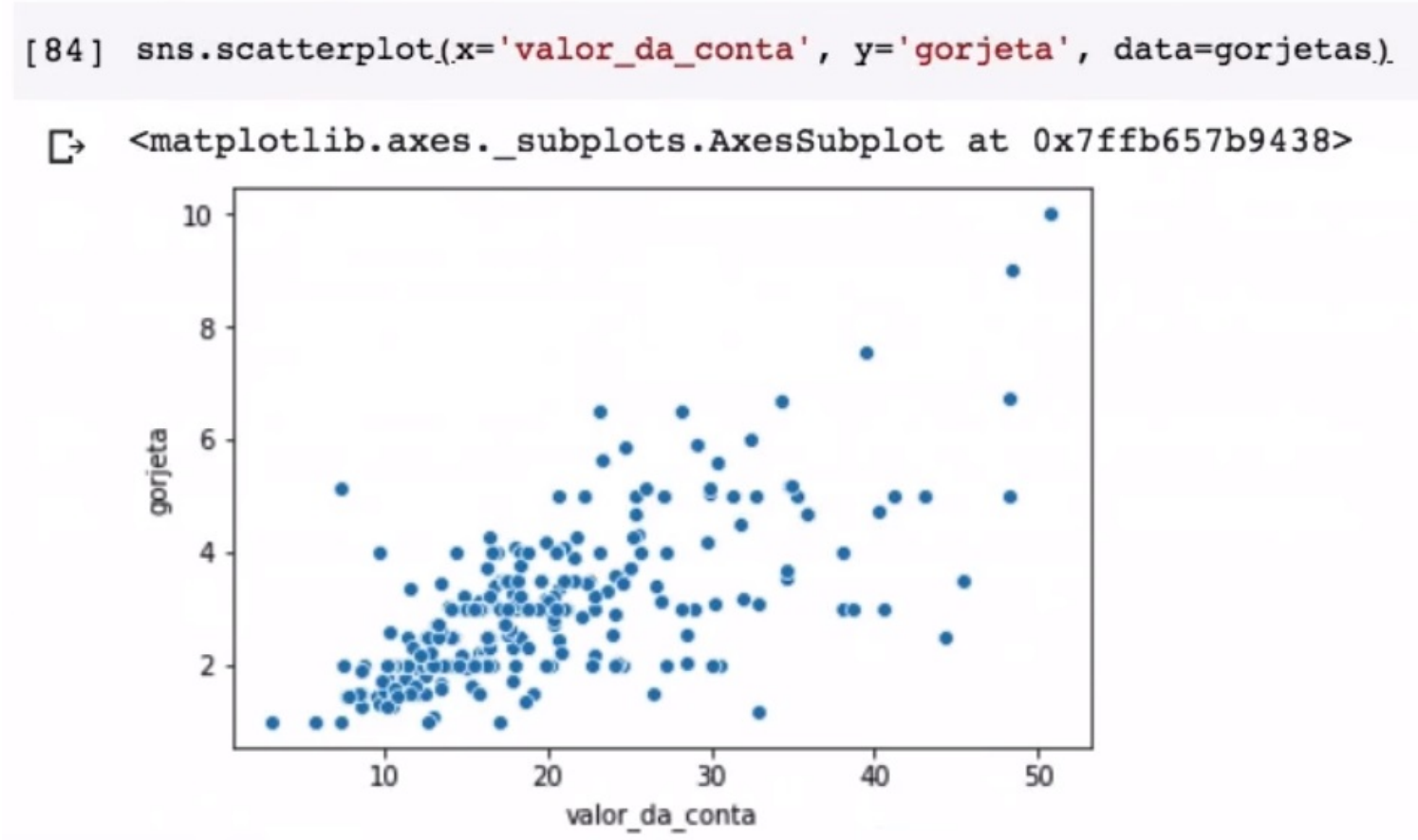
Chamaremos o Seaborn por `sns` e geraremos este primeiro gráfico com **`scatterplot()`** , o qual receberá três parâmetros: `x` referente ao valor do eixo `'valor_da_conta'` , `y` para `'gorjeta'` e a

base de dados `data-gorjetas` que estamos utilizando.

```
sns.scatterplot(x= 'valor_da_conta', y= 'gorjeta', data=gorjetas)
```

COPIAR CÓDIGO

Executando com "Shift + Enter", poderemos visualizar o novo gráfico. O sistema marcará o endereço de memória de onde esta imagem está.



Poderemos armazená-la em uma variável `valor_gorjeta` adicionando-a antes da sentença de `scatterplot`, fazendo com que o endereço desapareça.

```
valor_gorjeta = sns.scatterplot(x= 'valor_da_conta', y= 'gorjeta', data=gorjetas
```

COPIAR CÓDIGO

Analisando este primeiro gráfico, poderemos ver as relações entre os eixos `x` e `y`. Aparentemente, há uma progressão linear que indica o aumento do valor da gorjeta conforme o valor da conta é maior para cada mesa.

É importante sempre fazermos um comentário em uma célula textual seguinte sobre o que o gráfico apresenta visualmente.

Porém, existe uma questão interessante para nos atentarmos; é importante conhecermos bem nosso banco e sabermos que não existe nenhum valor nulo. Para sabermos quantos dados temos em nossa base, adicionaremos o comando `gorjetas.shape[0]`. Como retorno, o sistema nos apresentará o valor de **244** registros.

Para atestarmos se todos os registros estão realmente preenchidos ou são nulos, poderemos utilizar o `print()` para visualizarmos o relatório da exploração `'A base de dados contém {} registros`. Para formatar, usaremos `.format()` após a frase, recebendo `gorjetas.shape[0]`.

```
print('A base de dados contém {} registros'.format(gorjetas.shape[0]))
```

COPIAR CÓDIGO

Ao executar, veremos o sistema apontar a existência de 244 registros. Para descobrir quantos não são nulos, pularemos uma linha na mesma célula adicionando `/n` ao final da frase anterior e criaremos um novo `print()` para `'Registros não nulos`. Para fazer este cálculo, usamos o método `count` para `gorjetas`.

```
print('A base de dados contém {} registros \n'.format(gorjetas.shape[0]))
print('Registros não nulos')
gorjetas.count()
```

COPIAR CÓDIGO

Como recebemos o retorno do sistema apontando 244 registros para todos os campos, significa que todos estão preenchidos e não há nenhum dado nulo.

Portanto, nossa primeira visualização em gráfico está correta, e concluímos que aumentando o valor total da conta, o valor da gorjeta também tende a ser maior.

No próximo passo, faremos uma análise ainda mais interessante.