DATA ANALYTICS

FASE 1 | AULA 03 - ANÁLISE EXPLORATÓRIA DE DADOS

MANIPULAÇÃO DE DADOS

SUMÁRIO

O QUE VEM POR AÍ?

Fala analytics expert, tudo certo?!

Chegamos a mais um ponto importante em nossa jornada, onde iniciaremos o aprofundamento nas manipulações de dados.

Até o momento, você partiu do ambiente do Google Colab, onde construímos nossas primeiras visualizações acerca dos dados disponíveis do DATASUS.

E, se por um acaso, você não lembra direito o código exato, é só acessá-lo no GitHub e correr para o abraço!

Aqui você irá interagir com um conteúdo que passa por análises envolvendo índice, plots, aleatoriedade e série temporal em dataframes. É muito importante que sempre revise as aulas anteriores, realize os exercícios e bote em prática cada detalhe. Lembre-se que o hands on é fundamental! Além disso, nossos docentes e pessoa de Community Management estão disponíveis no Discord para te ajudar com possíveis dúvidas e apontamentos!

Agora vamos desbravar esse maravilhoso mundo dos dados!

HANDS ON

Agora, chegou o momento de ver, na prática, como começar a importar nossos dados e trabalhar com eles via programação. O ambiente utilizado é o Google Colab, e as bases de dados que foram disponibilizadas no início da disciplina. A ideia é não se limitar apenas ao código explicito no hands on, então procure a documentação das bibliotecas, explore novas funcionalidades e muito mais!

Confira as três partes da aula e faça as sugestões de desafios! O material completo você encontra no nosso GitHub.

SAIBA MAIS

Você viu anteriormente o poder que a biblioteca Pandas nos traz em questão de conteúdo para visualizar dados e as funções que facilitam a nossa vida.

Uma dessas vantagens é utilizar a tipagem dinâmica do Python para guardar os Dataframes com o que realmente importa. Outro ponto é perceber que a função ***.plot( )*** do Pandas nos permite trazer um gráfico sem rodar uma biblioteca gráfica previamente declarada com a palavra reservada “import”.

Mas vamos com calma, jovem padawan! É importante ressaltar que o uso de bibliotecas como o Seaborn, Matplotlib e Plotly ainda é importante! Elas são bibliotecas completas e que nos dão completude em análises gráficas, então não as descarte e analise com cuidado, já que o ***.plot( )*** do Pandas é excelente para análises rápidas dentro do seu Dataframe a ser analisado.

Outro ponto importante a ser analisado no tipo de base que estamos lidando é o fato da análise obrigatoriamente nos conduzir à um problema de cunho temporal, e isso será o princípio do que mais tarde chamaremos de séries temporais.

Outros pontos para uma análise importante em experimentos que a computação traz do método científico são a reprodutibilidade e aleatoriedade.

Você deve estar se perguntando: o que é isso? Então, anote aí no seu caderninho de palavras importantes! Quando falamos de uma análise de dados que tem como um dos seus pilares a reprodutibilidade, significa simplesmente que, se por um acaso você, ao analisar uma base, obter o resultado “X”, se outra pessoa em qualquer lugar do planeta quiser fazer a mesma coisa que você, ele deve obter o mesmo resultado “X”. Pois uma análise com resultados que não pode ser reproduzida por outra pessoa, é no mínimo suspeita e um tanto enviesada, não acha?

Faça uma analogia com um tema científico, por exemplo... Física! Imagine o tema “gravidade”, onde sabemos que os objetos são atraídos para o centro da terra, e por isso eles caem, mas os experimentos e constatação desse fenômeno físico devem ser iguais independentemente de onde você quiser testar no planeta Terra. Se por um acaso fosse impossível de reproduzir em todos os lugares, a teoria deveria ser revista, já que não seria válida para toda a situação.

Nos alongamos, mas é importante conceituar para que as palavras não se tornem enfeites sem um sentido aplicável. Mas... e a aleatoriedade?

Em um experimento, a aleatoriedade é importante para que, ao analisar os dados, independente do dado em questão, não nos apeguemos com especificidades, ou seja, a análise deve servir para qualquer valor.

Assista a segunda parte da aula, pois lá nós trabalhamos a temática de uma maneira muito bacana!

Para chegarmos ao último ponto importante em nossas análises, se faz necessário falar sobre a ordenação dos dados analisados.

Talvez você esteja se perguntando: mas por que eu deveria organizar as coisas, já que antes foram mencionados os dados aleatórios?

A resposta é simples: há uma diferença entre ter dados aleatórios e dados desorganizados!

Se você quer analisar variações do preço de 7 tipos de ações na bolsa de valores ao longo dos últimos 4 anos, pensando que você pode ter uma carteira com 100 ações, é importante que uma amostra (sample) de 7 dessas ações seja solicitada. E como seu modelo deve valer para qualquer uma, você pode pegá-las aleatoriamente, mas ainda sim é importante que, enquanto um especialista na área, você saiba ordenar isso temporalmente para enxergar com clareza toda a situação.

Isso significa que a escolha dos dados para serem analisados pode ser aleatória, mas é interessante que você os ordene conforme sua necessidade e problema de negócio.

No Python, existe uma solução dentro da biblioteca Numpy, onde chamamos uma linha de código antes de começar o resto do programa:

```
np.random.seed(valor)
```

Onde essa “semente” garante um estado de aleatoriedade salvo, ou seja, se chamarmos essa função com um valor fixo, várias vezes, o computador exibirá os mesmos números aleatórios.

Mas existe um ponto de alerta! Usar isso globalmente pode ser ruim em projetos grandes, já que isso pode afetar o módulo numpy.random.* como um todo.

O ideal é que criemos um objeto gerador com essa geração e utilizemos na medida do possível. Essa convenção foi criada por Robert Kern e se encontra em uma normativa da documentação do Numpy chamada NEP 19 <https://numpy.org/neps/nep-0019-rng-policy.html>.

Mas, para entendermos melhor a ideia global, imagine que você tenha o seguinte trecho de código:

```
import numpy as np

np.random.seed(1)
array = np.random.rand(5)
np.random.seed(1)
array2 = np.random.rand(5)
print(array)
print(array2)
```

Como nosso seed foi repetido entre as duas variáveis array, é de se esperar que encontremos um valor igual:

```
[4.17022005e-01 7.20324493e-01 1.14374817e-04 3.02332573e-01
 1.46755891e-01]
[4.17022005e-01 7.20324493e-01 1.14374817e-04 3.02332573e-01
 1.46755891e-01]
```

Aqui está bem evidente a garantia de reprodutibilidade. Contudo, olhemos agora para uma questão onde apenas chamamos o seed uma vez:

```
import numpy as np

np.random.seed(1)
array = np.random.rand(5)

array2 = np.random.rand(5)
print(array)
print(array2)
```

Em seguida temos output:

```
[4.17022005e-01 7.20324493e-01 1.14374817e-04 3.02332573e-01
 1.46755891e-01]
[0.09233859 0.18626021 0.34556073 0.39676747 0.53881673]
```

No primeiro array era de se esperar que mantivéssemos o valor do bloco de código anterior, mas veja que a variável array2 assume um novo valor, já que pela lei de formação de dados pseudoaleatórios, o valor será diferente, porque a aleatoriedade se garantiu apenas na primeira chamada após o código.

Os computadores são incapazes de gerar um número verdadeiramente aleatório, porque os computadores são determinísticos e seguem consistentemente um determinado conjunto de instruções. A ideia por trás disso é que sempre obteremos o mesmo conjunto de números aleatórios para a mesma semente em qualquer máquina.

Historicamente, temos algoritmos geradores de números pseudoaleatórios, desde John von Neumann, em 1946, até um algoritmo (Itamaracá) criado por um brasileiro chamado Daniel Henrique Pereira, em 2021.

O mais importante nisso tudo é ter em mente que o processo aleatório em máquinas jamais será 100%, já que o próprio hardware é determinístico. Em contrapartida, a pseudoaleatoriedade nos traz os benefícios de algo que podemos gerar reprodutibilidade, sem nos preocuparmos se garantiremos isso de forma escalável.

Dica de conteúdo:

Para entender a geração de números aleatórios, veja o primeiro <https://www.youtube.com/watch?v=p5-uRvEK5WE> e o segundo **<https://www.youtube.com/watch?v=NuQIJcEOjBY>** vídeo de uma aula show sobre o tema!

O QUE VOCÊ VIU NESTA AULA

Você viu como os números pseudoaleatórios são gerados e como obter uma amostra aleatória dos dados. Além disso, aprendeu a como manipular os dados para plotar informações dos estados de desejo, por meio de ordenação, e o que são séries temporais.

IMPORTANTE: não esqueça de praticar com o desafio da disciplina, para que assim você possa aprimorar os seus conhecimentos!

Você não está sozinho ou sozinha nesta jornada! Te esperamos no Discord e nas lives com os especialistas, onde você poderá tirar dúvidas, compartilhar conhecimentos e estabelecer conexões!

REFERÊNCIAS

DATASUS. **<https://datasus.saude.gov.br/>.** Acesso em: 07 fev. 2023.

DOCUMENTAÇÃO PANDAS. **<https://pandas.pydata.org/>.** Acesso em: 07 fev. 2023.

GOOGLE COLAB. **<https://colab.research.google.com/>.** Acesso em: 07 fev. 2023.

PEREIRA, D. H. Itamaracá: A Novel Simple Way to Generate Pseudo-random Numbers. <https://www.cambridge.org/engage/api-gateway/coe/assets/orp/resource/item/61b410fadcbea24f839f0235/original/itamaraca-a-novel-simple-way-to-generate-pseudo-random-numbers.pdf>. Acesso em: 07 fev 2023.

TABNET. **<https://datasus.saude.gov.br/informacoes-de-saude-tabnet/>.** Acesso em: 07 fev. 2023.

PALAVRAS-CHAVE

Python. Pandas. Dataframe.