**11 - Estatísticas com NumPy**

Para finalizarmos essa seção de treinamento básico de Numpy, aprenderemos como obter estatísticas descritivas usando os arrays Numpy e alguns métodos simples. Claro, existem outros métodos disponibilizados por esse pacote e que podem ser encontrados na documentação.

No começo do curso, fizemos o upload dos arquivos para o Colab. Caso você tenha fechado o notebook alguma vez, será necessário fazer esse processamento novamente utilizando a aba "Files" à esquerda. Criaremos, então, três variáveis que receberão arrays contendo os dados dos conjuntos carros-anos.txt, carros-km.txt e carros-valor.txt.

Para isso, usaremos o já conhecido método np.loadtxt() passando como parâmetro fname, que define o nome do arquivo a ser carregado. Além disso, no carregamento de carros-anos.txt, usaremos o parâmetro dtype = int somente para ilustrarmos a definição de tipos de dados.

anos = np.loadtxt(fname = "carros-anos.txt", dtype = int)

km = np.loadtxt(fname = "carros-km.txt")

valor = np.loadtxt(fname = "carros-valor.txt")COPIAR CÓDIGO

Usando o atributo shape, conseguiremos verificar que anos é um array de uma dimensão com 258 linhas.

anos.shapeCOPIAR CÓDIGO

(258, )

Queremos agora juntar os nossos arrays, e o Numpy nos oferece uma funcionalidade que permitirá fazermos isso de maneira bem simples: o método **column\_stack()**, que transforma arrays unidimensionais em colunas de um array bidimensional.

Criaremos então uma variável dataset para qual atribuiremos a chamada de np.column\_stack() recebendo como parâmetro uma tupla contendo os arrays anos, km e valor.

dataset = np.column\_stack((anos, km, valor))

datasetCOPIAR CÓDIGO

Como resultado, teremos um único array cujas colunas representam cada um dos arrays anteriores.

array([[2.0030000e+03, 4.4410000e+04, 8.8078640e+04], [1.9910000e+03, 5.7120000e+03, 1.0616194e+05], [1.9900000e+03, 3.7123000e+04, 7.2832160e+04], [2.0190000e+03, 0.0000000e+00, 1.2454907e+05], [2.0060000e+03, 2.5757000e+04, 9.2612100e+04], ...])

Entretanto, note que, como os arrays km e valor são representados em float, os anos também passaram a ser listados nesse tipo. Com dataset.shape, podemos verificar que este é um array único de duas dimensões com 258 linhas e 3 colunas.

dataset.shapeCOPIAR CÓDIGO

(258, 3)

A primeira estatística que calcularemos usando esse dataset é a media, que conseguiremos usando o método np.mean().

np.mean(dataset)COPIAR CÓDIGO

Ao executarmos esse código, teremos como retorno um único número:

48489.14648578811

Isso acontece porque o Numpy pegou todos os valores do conjunto indiscriminadamente - ou seja, estamos calculando a média de todos os anos, quilometragens e valores juntos, algo que não faz nenhum sentido. Precisaremos informar ao Numpy com que eixo queremos trabalhar, as linhas ou as colunas, algo que é possível por meio do parâmetro axis. Com axis=0, por exemplo, conseguiremos a média das colunas.

np.mean(dataset, axis = 0)COPIAR CÓDIGO

array([ 2007.51162791, 44499.41472868, 98960.51310078])

Note que a média dos anos não faz sentido, mas a da quilometragem e a do valor sim. Se os dados estivessem organizados em linhas (por exemplo, com cada linha contendo informações sobre um veículo), usaríamos axis=1.

Como não faz sentido calcularmos a média dos anos de fabricação, podemos simplesmente fazer um fatiamento para mantermos todas as linhas e somente a coluna da quilometragem ou do valor.

p.mean(dataset[:,1])COPIAR CÓDIGO

44830.54666666667

np.mean(dataset[:,2])COPIAR CÓDIGO

41392.763333333336

Além da média, podemos calcular o desvio padrão dos valores usando o método **np.std()**.

np.std(dataset[:,2])COPIAR CÓDIGO

29754.101150388564

*Não entraremos em detalhes sobre o que representam a média ou o desvio padrão, já que esse não é um curso focado em estatística. Se estiver interessado(a), pode conferir outros cursos de estatística e análise de dados aqui mesmo na Alura!*

A função **sum()** do Numpy nos retorna a somatória dos valores do array. Claro, para que esse cálculo faça sentido, é necessário passarmos o eixo com o qual queremos trabalhar.

dataset.sum(axis = 0)COPIAR CÓDIGO

array([ 517938. , 11480849. , 25531812.37999999])

Também podemos obter somente a soma das quilometragens, por exemplo, fatiando o dataset de modo que ele só exiba a segunda coluna:

dataset[:, 1].sum()COPIAR CÓDIGO

11480849.0

Outra possibilidade é fazermos esse cálculo usando o método np.sum() e passando o dataset como parâmetro.

np.sum(dataset, axis = 0)COPIAR CÓDIGO

array([ 517938. , 11480849. , 25531812.37999999])

Essa sintaxe também permite o fatiamento das informações que desejamos apresentar.

np.sum(dataset[:, 2])COPIAR CÓDIGO

25531812.38

Esse é o conteúdo introdutório sobre Numpy que planejamos para essa primeira etapa do treinamento de ciência de dados com o Python. Na segunda parte do curso, conversaremos também sobre o pacote Pandas, e você perceberá algumas semelhanças entre as funcionalidades desses pacotes - afinal, o Pandas foi construído em cima do pacote Numpy.

No próximo vídeo faremos o encerramento do curso!