

Fundamentos de Data Science para Finanças

Prof. Dr. Daniel Bergmann

Python para Data Science

Introdução

Olá, sou o Professor Daniel Bergmann do curso de Machine Learning e Data Science com Python do LIT. Neste tópico, aprenderemos as principais aplicações dos pacotes Pandas, Numpy e Matplotlib na arquitetura e formulação da análise de dados. É recomendável que os participantes e aluno(a)s já tenham assistido a unidade sobre Fundamentos de Python.

Espero que seja uma ótima experiência de aprendizado e que possamos vencer todos os desafios juntos. Neste módulo, vamos dar nossos primeiros passos na ciência de dados com os principais pacotes do Python.

Introdução ao Numpy

Nesta primeira seção desta apostila sobre Python para Data Science conversaremos sobre o pacote Numpy, uma abreviação de Numerical Python. Este é um dos pacotes mais importantes para processamento numérico no Python, além de ser a base para a maioria dos pacotes científicos que utilizam dados numéricos. No notebook deixamos inclusive um texto explicativo sobre alguns detalhes do pacote.

Alguns pontos importantes são a disponibilidade de um poderoso objeto array multidimensional, as funções matemáticas sofisticadas para operações com arrays sem a necessidade de utilização de laços 'for', e os recursos de álgebra linear e de geração de números aleatórios. O Numpy também serve como um contêiner eficiente para transporte de dados multidimensionais entre algoritmos e pacotes com os quais trabalhamos, além de servir de base para construção de outros pacotes.

Existem diversos pacotes do Python que não são distribuídos na versão padrão (default) do seu interpretador, como o próprio Numpy, o Pandas, o Scikit-learn, o Matplotlib e o Seaborn. O Google Colab disponibiliza esses pacotes previamente, eliminando a necessidade de sua instalação. Caso você esteja usando outra distribuição, a instalação pode ser necessária.

Para utilizarmos os métodos e funções de interesse em cada pacote, é necessário importá-los. No caso do Numpy, basta usarmos a palavra-chave import (reservada do Python) seguida do nome dessa biblioteca.

```
>> import numpy
```

Na documentação do Numpy, encontramos algumas explicações sobre o `arange()`, que basicamente funciona similarmente como a função `range()`, ou seja, ela cria uma lista (nesse caso, um array numpy) que se inicia no 0 e com tamanho indicado pelo próprio parâmetro.

Para chamarmos essa função, usaremos `numpy.arange()`, passando como valor o número 10. Como retorno, teremos um array de tamanho 10:

```
>> numpy.arange(10)
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Existe também outra forma de fazer essa importação, que é atribuindo um apelido à biblioteca, algo que fizemos no início em Fundamentos de Python.

```
>> import numpy as np
```

Esse tipo de importação é muito comum em livros e textos sobre Data Science, e se estende também a outras bibliotecas. O pacote Pandas, por exemplo, costuma ser apelidada de pd. Com a atribuição do apelido, poderemos chamar a função `arange()` da seguinte forma:

```
>> np.arange(10)
```

Agora começaremos a criar propriamente os famosos ‘arrays Numpy’ ou vetores. Uma das maneiras de fazermos isso é por meio de uma lista. Para isso, criaremos uma variável `km` que receberá a chamada de `np.array()`. Passaremos como parâmetro dessa função uma lista contendo números aleatórios.

```
>> km = np.array([1000, 2300, 4987, 1500])
km
```

```
array([1000, 2300, 4987, 1500])
```

Existem também arrays com mais de uma dimensão. Nesse caso, trabalharemos apenas com arrays de duas dimensões, deixando as outras possibilidades para futuros exemplos e casos. Esse tipo de array funciona como uma matriz ou um dataset. De modo a facilitarmos nosso aprendizado, usaremos a seguinte lista pré-preparada.

```
>> dados = [
    ['Rodas de liga', 'Travas elétricas', 'Piloto automático', 'Bancos de couro',
     'Ar condicionado', 'Sensor de estacionamento', 'Sensor crepuscular', 'Sensor de chuva'],
    ['Central multimídia', 'Teto panorâmico', 'Freios ABS', '4 X 4', 'Painel digital', 'Piloto automático', 'Bancos de couro', 'Câmera de estacionamento'],
    ['Piloto automático', 'Controle de estabilidade', 'Sensor crepuscular', 'Freios ABS', 'Câmbio automático', 'Bancos de couro', 'Central multimídia', 'Vidros elétricos']
]
dados
```

Criaremos então uma variável ‘Acessorios’ à qual atribuiremos a chamada de `np.array()`. Esta, por sua vez, receberá por parâmetro a lista denominada `dados` acima.

```
>> Acessorios = np.array(dados)
Acessorios
```

Com isso, teremos um array numpy contendo toda a nossa lista.

```
>> array([[ 'Rodas de liga', 'Travas elétricas', 'Piloto automático',
           'Bancos de couro', 'Ar condicionado', 'Sensor de estacionamento',
           'Sensor crepuscular', 'Sensor de chuva'],
          ['Central multimídia', 'Teto panorâmico', 'Freios ABS', '4 X 4',
           'Painel digital', 'Piloto automático', 'Bancos de couro',
           'Câmera de estacionamento'],
          ['Piloto automático', 'Controle de estabilidade',
```

```
'Sensor crepuscular', 'Freios ABS', 'Câmbio automático',  
'Bancos de couro', 'Central multimídia', 'Vidros elétricos']],  
dtype='<U24')
```

É possível verificarmos a dimensão de cada um dos arrays que acabamos de criar utilizando a instrução `shape`. Começaremos com `km.shape`:

```
>> km.shape
```

```
(258,)
```

Isso significa que o nosso array possui 258 "linhas" e uma única dimensão. Mas observe o que acontece quando fazemos o mesmo para 'Acessorios':

```
>> Acessorios.shape
```

```
(3,8)
```

Esse retorno é uma dupla indicando que o array possui 3 linhas e 8 colunas.

Agora que conhecemos um pouco do funcionamento desses arrays, ainda falta entendermos o motivo para utilizá-los ao invés das listas. Primeiramente, os arrays têm um desempenho muito melhor que o das listas quando fazemos operações matemáticas com eles.

Começaremos trabalhando com as listas para exemplificar as principais operações com Numpy

```
>> km = [44410., 5712., 37123., 0., 25757.]  
anos = [2003, 1991, 1990, 2019, 2006]
```

Essas listas são ordenadas, ou seja, o primeiro valor de cada uma corresponde ao primeiro carro; o segundo, ao segundo carro; e assim sucessivamente. Queremos fazer um cálculo simples para descobrir a idade do veículo, algo que conseguiremos subtraindo o ano mais recente (2019, por hipótese) pelo ano de fabricação de cada automóvel.

Criaremos então uma variável `idade` que receberá a subtração de 2019 pela lista 'anos'.

```
>> idade = 2019 - anos
```

A execução desse código nos retornará um erro, já que não é possível fazer a subtração de um inteiro por uma lista. Entretanto, isso é possível com os arrays Numpy.

```
>> km = np.array([44410., 5712., 37123., 0., 25757.])  
anos = np.array([2003, 1991, 1990, 2019, 2006])
```

Ao subtrairmos 2019 pelo array `anos`, o retorno será outro array Numpy que poderá ser exibido na tela, este contendo as idades de cada um dos veículos.

```
>> idade = 2019 - anos  
idade
```

```
array([16, 28, 29, 0, 13])
```

Também podemos realizar operações usando dois ou mais arrays. Por exemplo, se quisermos a quilometragem média (km_media) podemos simplesmente dividir o array km pela variável idade que acabamos de criar.

```
>> km_media = km / idade
```

Nesse caso teremos uma saída que indica um erro, já que nosso conjunto contém um valor nulo, incorrendo em uma divisão por zero. Entretanto, o próprio Numpy preenche esse valor com NaN (*not a number*) e continua a operação.

```
array([2775.625 , 204. , 1280.10344828, nan, 1981.30769231])
```

Perceba que não tivemos que utilizar nenhum tipo de laço ou criar outros arrays para realizarmos essas contas, que afinal são bem simples. Nesse momento estamos trabalhando apenas com uma dezena de dados, mas imagine se tivéssemos milhões? Levaria muito tempo para percorrermos todas as informações e realizarmos as operações usando laços, e isso custaria muito desempenho.

Para mostra um array multidimensional, criaremos o novo array dados usando como base os arrays km e anos que criamos anteriormente.

```
>> dados = np.array([km, anos])
dados
```

Como retorno, temos um array Numpy com duas linhas e 5 colunas (2,5), que representando uma matriz. Isso é ilustrado pelo desenho abaixo, lembrando que a indexação se inicia no 0:

```

      0      1      2      3      4
      ↓      ↓      ↓      ↓      ↓
array([[ 44410,  5712,  37123,    0, 25757 ], ← 0
       [ 2003, 1991,  1990, 2019,  2006 ]]) ← 1

```

Agora queremos acessar esses dados de modo a calcularmos novamente a quilometragem média. Da mesma forma que fizemos com as listas, o índice 0 indica que estamos acessando o primeiro item, ou seja, o primeiro array/linha contendo 5 dados. Já com dados[1], acessaremos o segundo índice/linha. Ou seja, para obtermos km_media, podemos simplesmente dividir dados[0], que representa a quilometragem, por 2019 - dados[1], que é a idade dos veículos.

```
>> km_media = dados[0] / (2019 - dados[1])
km_media
```

Receberemos um aviso indicando que existe uma divisão por zero, mas a lista com os resultados será criada normalmente:

```
array([2775.625 , 204. , 1280.10344828, nan, 1981.30769231])
```

Introdução ao Pacote Pandas

Pandas é uma ferramenta de manipulação de dados de alto nível, construída com base no pacote Numpy. O pacote pandas possui estruturas de dados bastante interessantes para manipulação de dados e por isso é muito utilizado por cientistas de dados.

Suponhamos um dataset contendo dados sobre características ou atributos de preços de veículos em uma concessionária do Estado de São Paulo.

	Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
0	Jetta Variant	Motor 4.0 Turbo	2003	44410.0	False	['Rodas de liga', 'Travas elétricas', 'Piloto ...	88078.64
1	Passat	Motor Diesel	1991	5712.0	False	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94
2	Crossfox	Motor Diesel V8	1990	37123.0	False	['Piloto automático', 'Controle de estabilidad...	72832.16
3	DS5	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétrico...	124549.07
4	Aston Martin DB4	Motor 2.4 Turbo	2006	25757.0	False	['Rodas de liga', '4 X 4', 'Central multimídia...	92612.10
...
253	Phantom 2013	Motor V8	2014	27505.0	False	['Controle de estabilidade', 'Piloto automátic...	51759.58
254	Cadillac Ciel concept	Motor V8	1991	29981.0	False	['Bancos de couro', 'Painel digital', 'Sensor ...	51667.06
255	Classe GLK	Motor 5.0 V8 Bi-Turbo	2002	52637.0	False	['Rodas de liga', 'Controle de tração', 'Câmbi...	68934.03
256	Aston Martin DB5	Motor Diesel	1996	7685.0	False	['Ar condicionado', '4 X 4', 'Câmbio automátic...	122110.90
257	Macan	Motor Diesel V6	1992	50188.0	False	['Central multimídia', 'Teto panorâmico', 'Vid...	90381.47

258 rows x 7 columns

Trata-se de um DataFrame do Pandas com 258 linhas e 7 colunas, compreendendo os atributos dos veículos (Nome, Motor, Ano, Kilometragem, Zero_Km, Acessórios e Valor).

Para calcular a estatística descritiva dos Valores dos veículos, podemos usar o seguinte comando:

```
>> dataset['Valor'].describe()
```

```
count      258.000000
mean      98960.513101
std       29811.932305
min       50742.100000
25%       70743.512500
50%       97724.380000
75%      124633.302500
max      149489.920000
Name: Valor, dtype: float64
```

Series são arrays unidimensionais rotulados capazes de armazenar qualquer tipo de dado. Os rótulos das linhas são chamados de index. A forma básica de criação de uma Series é a seguinte:

```
s = pd.Series(dados, index = index)
```

O argumento *dados* pode ser um dicionário, uma lista, um array Numpy ou uma constante.

Já um DataFrame é uma estrutura de dados tabular bidimensional com rótulos nas linhas e colunas. Como a Series, os DataFrames são capazes de armazenar qualquer tipo de dados.

```
df = pd.DataFrame(dados, index = index, columns = columns)
```

O argumento *dados* pode ser um dicionário, uma lista, um array Numpy, uma Series e outro DataFrame. Fonte da Documentação: <https://pandas.pydata.org/pandas-docs/version/0.25/>

Podemos criar uma Série a partir de uma lista como no exemplo a seguir:

```
>> carros = ['Jetta Variant', 'Passat', 'Crossfox']
carros

>> pd.Series(carros)

0    Jetta Variant
1         Passat
2       Crossfox
dtype: object
```

Podemos também criar um DataFrame a partir de uma lista de dicionários conforme o exemplo a seguir:

```
>> dados = [
    {'Nome': 'Jetta Variant', 'Motor': 'Motor 4.0 Turbo', 'Ano': 2003, 'Quilometragem':
44410.0, 'Zero_km': False, 'Valor': 88078.64},
    {'Nome': 'Passat', 'Motor': 'Motor Diesel', 'Ano': 1991, 'Quilometragem': 5712.0,
'Zero_km': False, 'Valor': 106161.94},
    {'Nome': 'Crossfox', 'Motor': 'Motor Diesel V8', 'Ano': 1990, 'Quilometragem':
37123.0, 'Zero_km': False, 'Valor': 72832.16}
]
>> dataset = pd.DataFrame(dados)
```

	Nome	Motor	Ano	Quilometragem	Zero_km	Valor
0	Jetta Variant	Motor 4.0 Turbo	2003	44410.0	False	88078.64
1	Passat	Motor Diesel	1991	5712.0	False	106161.94
2	Crossfox	Motor Diesel V8	1990	37123.0	False	72832.16

Ou ainda, a partir de um dicionário:

```
>> dados = {
    'Nome': ['Jetta Variant', 'Passat', 'Crossfox'],
    'Motor': ['Motor 4.0 Turbo', 'Motor Diesel', 'Motor Diesel V8'],
    'Ano': [2003, 1991, 1990],
    'Quilometragem': [44410.0, 5712.0, 37123.0],
    'Zero_km': [False, False, False],
    'Valor': [88078.64, 106161.94, 72832.16]}
```

```
>> dataset = pd.DataFrame(dados)
```

	Nome	Motor	Ano	Quilometragem	Zero_km	Valor
0	Jetta Variant	Motor 4.0 Turbo	2003	44410.0	False	88078.64
1	Passat	Motor Diesel	1991	5712.0	False	106161.94
2	Crossfox	Motor Diesel V8	1990	37123.0	False	72832.16

Para selecionar a coluna 'Valor' basta digitar o código:

```
>> dataset['Valor']
```

```
Nome
Jetta Variant      88078.64
Passat             106161.94
Crossfox           72832.16
DS5                124549.07
Aston Martin DB4   92612.10
...
Phantom 2013       51759.58
Cadillac Ciel concept 51667.06
Classe GLK         68934.03
Aston Martin DB5   122110.90
Macan              90381.47
Name: Valor, Length: 258, dtype: float64
```

Selecionando linhas - [i : j]

Observação: A indexação tem origem no zero e nos fatiamentos (*slices*) a linha com índice i é **incluída** e a linha com índice j **não é incluída** no resultado.

Assim, para selecionar as três primeiras linhas do banco de dados, digitamos:

```
dataset[0:3]
```

	Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
	Jetta Variant	Motor 4.0 Turbo	2003	44410.0	False	['Rodas de liga', 'Travas elétricas', 'Piloto ...	88078.64
	Passat	Motor Diesel	1991	5712.0	False	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94
	Crossfox	Motor Diesel V8	1990	37123.0	False	['Piloto automático', 'Controle de estabilidade...	72832.16

Utilização do método loc[] para seleções

Observação: Seleciona um grupo de linhas e colunas segundo os rótulos ou uma matriz booleana. A função loc[] tem o intuito de localizar linhas e colunas com base nos rótulos do objeto. Para selecionar as características do carro Passat (dados que os nomes dos carros estão no índice do Data Frame), devemos escrever:


```
dataset.loc['Passat']
```

```
Motor          Motor Diesel
Ano            1991
Quilometragem  5712
Zero_km        False
Acessórios     ['Central multimídia', 'Teto panorâmico', 'Fre...
Valor          106162
Name: Passat, dtype: object
```

Caso desejarmos selecionar dois tipos de carros, procederemos da seguinte forma:

```
dataset.loc[['Passat', 'DS5']]
```

	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
Nome						
Passat	Motor Diesel	1991	5712.0	False	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94
DS5	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétrico...	124549.07

```
dataset.loc[['Passat', 'DS5'], ['Motor', 'Valor']]
```

	Motor	Valor
Nome		
Passat	Motor Diesel	106161.94
DS5	Motor 2.4 Turbo	124549.07

Para selecionar todos os carros pelas colunas declaradas como 'Motor' e 'Valor' colocamos o símbolo (:) na primeira entrada da função loc, como se pode ver logo abaixo:

```
dataset.loc[:, ['Motor', 'Valor']]
```

	Motor	Valor
Nome		
Jetta Variant	Motor 4.0 Turbo	88078.64
Passat	Motor Diesel	106161.94
Crossfox	Motor Diesel V8	72832.16
DS5	Motor 2.4 Turbo	124549.07
Aston Martin DB4	Motor 2.4 Turbo	92612.10
...
Phantom 2013	Motor V8	51759.58
Cadillac Ciel concept	Motor V8	51667.06
Classe GLK	Motor 5.0 V8 Bi-Turbo	68934.03
Aston Martin DB5	Motor Diesel	122110.90
Macan	Motor Diesel V6	90381.47

258 rows x 2 columns

Utilização do método `iloc[]` para seleções

Observação: Selecciona com base nos índices, ou seja, se baseia na posição das informações. A função `iloc[]` tem o intuito de localizar linhas e colunas com base nos índices de posição do objeto.

Olhando nosso dataset novamente por meio do método `head()`

```
dataset.head()
```

	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
Nome						
Jetta Variant	Motor 4.0 Turbo	2003	44410.0	False	['Rodas de liga', 'Travas elétricas', 'Piloto ...	88078.64
Passat	Motor Diesel	1991	5712.0	False	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94
Crossfox	Motor Diesel V8	1990	37123.0	False	['Piloto automático', 'Controle de estabilidade...	72832.16
DS5	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétrico...	124549.07
Aston Martin DB4	Motor 2.4 Turbo	2006	25757.0	False	['Rodas de liga', '4 X 4', 'Central multimídia...	92612.10

Para seleccionar a segunda linha utilizaremos o número 1 dentro da função `iloc[]`

```
dataset.iloc[[1]]
```

	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
Nome						
Passat	Motor Diesel	1991	5712.0	False	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94

Para seleccionar as linhas que compreendem os carros (Passat, Crossfox e DS5) cujos índices do DataFrame são (1,2,3), escrevemos:

```
dataset.iloc[1:4]
```

	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
Nome						
Passat	Motor Diesel	1991	5712.0	False	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94
Crossfox	Motor Diesel V8	1990	37123.0	False	['Piloto automático', 'Controle de estabilidade...	72832.16
DS5	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétrico...	124549.07

Para capturar do DataFrame acima a sequência de colunas (Motor, Valor, Quilometragem), digitamos [0,5,2] no campo das colunas do método do `iloc`.

```
dataset.iloc[1:4, [0, 5, 2]]
```

Nome	Motor	Valor	Quilometragem
Passat	Motor Diesel	106161.94	5712.0
Crossfox	Motor Diesel V8	72832.16	37123.0
DS5	Motor 2.4 Turbo	124549.07	NaN

Para pegar todos os carros nas linhas, devemos digitar:

```
dataset.iloc[:, [0, 5, 2]]
```

Nome	Motor	Valor	Quilometragem
Jetta Variant	Motor 4.0 Turbo	88078.64	44410.0
Passat	Motor Diesel	106161.94	5712.0
Crossfox	Motor Diesel V8	72832.16	37123.0
DS5	Motor 2.4 Turbo	124549.07	NaN
Aston Martin DB4	Motor 2.4 Turbo	92612.10	25757.0
...
Phantom 2013	Motor V8	51759.58	27505.0
Cadillac Ciel concept	Motor V8	51667.06	29981.0
Classe GLK	Motor 5.0 V8 Bi-Turbo	68934.03	52637.0
Aston Martin DB5	Motor Diesel	122110.90	7685.0
Macan	Motor Diesel V6	90381.47	50188.0

Realização de Queries com DataFrames

Imaginemos que gostássemos de selecionar os carros com Motor Diesel e Zero Km. Para tanto, podemos criar o seguinte conjunto booleano (verdadeiros e falsos) e colocar dentro do dataset original

```
Filtro = (dataset.Motor == 'Motor Diesel') & (dataset.Zero_km == True)
>> dataset[Filtro]
```

Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
Sorento	Motor Diesel	2019	NaN	True	['Sensor de chuva', 'Câmera de estacionamento'...	81399.35
Cielo Hatch	Motor Diesel	2019	NaN	True	['Painel digital', 'Central multimídia', 'Câme...	145197.70
Camry	Motor Diesel	2019	NaN	True	['Travas elétricas', 'Rodas de liga', 'Sensor ...	138597.27
Aston Martin Virage	Motor Diesel	2019	NaN	True	['Travas elétricas', 'Controle de tração', 'Câ...	97290.18
Série 7 Sedã	Motor Diesel	2019	NaN	True	['Vidros elétricos', 'Travas elétricas', 'Roda...	67539.79

Pelo método query(), teríamos:

```
dataset.query('Motor == "Motor Diesel" and Zero_km == True')
```

Podemos também fazer iterações nas linhas do DataFrame usando o comando `for()`. Utilizaremos os seguintes métodos ou funções:

- Método `iterrows()` – gera os índices e as linhas para fazer a seleção e procedimentos no DataFrame. No caso o index representará o nome dos carros dentro do nosso dataset e row será as linhas do dataset
- Index será usado dentro da função `loc[]` por conter nomes de rótulos dos carros e `row['Quilometragem']` representará o valor da Quilometragem em relação ao respectivo carro (index)
- Construiremos a coluna `Km_media` com base na Quilometragem do Carro dividido pelo número de anos do veículo até 2019 (data atual).

```
for index, row in dataset.iterrows():
    if(2019 - row['Ano'] != 0):
        dataset.loc[index, 'Km_media'] = row['Quilometragem'] / (2019 - row['Ano'])
    else:
        dataset.loc[index, 'Km_media'] = 0
```

dataset

Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor	Km_media
Jetta Variant	Motor 4.0 Turbo	2003	44410.0	False	['Rodas de liga', 'Travas elétricas', 'Piloto ...	88078.64	2775.625000
Passat	Motor Diesel	1991	5712.0	False	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94	204.000000
Crossfox	Motor Diesel V8	1990	37123.0	False	['Piloto automático', 'Controle de estabilidade...	72832.16	1280.103448
DS5	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétrico...	124549.07	0.000000
Aston Martin DB4	Motor 2.4 Turbo	2006	25757.0	False	['Rodas de liga', '4 X 4', 'Central multimídia...	92612.10	1981.307692
...
Phantom 2013	Motor V8	2014	27505.0	False	['Controle de estabilidade', 'Piloto automátic...	51759.58	5501.000000
Cadillac Ciel concept	Motor V8	1991	29981.0	False	['Bancos de couro', 'Painel digital', 'Sensor ...	51667.06	1070.750000
Classe GLK	Motor 5.0 V8 Bi-Turbo	2002	52637.0	False	['Rodas de liga', 'Controle de tração', 'Câmbi...	68934.03	3096.294118
Aston Martin DB5	Motor Diesel	1996	7685.0	False	['Ar condicionado', '4 X 4', 'Câmbio automátic...	122110.90	334.130435
Macan	Motor Diesel V6	1992	50188.0	False	['Central multimídia', 'Teto panorâmico', 'Vid...	90381.47	1858.814815

258 rows x 7 columns

Tratamento de Dados

Para selecionar os dados faltantes NaN da coluna Quilometragem, podemos proceder:

```
Filtro = dataset['Quilometragem'] #Foi selecionado Quilometragem
```

```
>> dataset[Filtro].isna() #Seleciona os dados faltantes
```

	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor	Km_media
Nome							
DS5	Motor 2.4 Turbo	2019	NaN	True	['Travas elétricas', '4 X 4', 'Vidros elétrico...	124549.07	0.0
A5	Motor 4.0 Turbo	2019	NaN	True	['Câmbio automático', 'Câmera de estacionamento...	56445.20	0.0
J5	Motor V6	2019	NaN	True	['Sensor crepuscular', 'Painel digital', 'Roda...	53183.38	0.0
A3	Motor 1.0 8v	2019	NaN	True	['4 X 4', 'Piloto automático', 'Central multim...	88552.39	0.0
Série 1 M	Motor V8	2019	NaN	True	['Controle de estabilidade', 'Central multimíd...	94564.40	0.0
...
Lamborghini Reventón	Motor 4.0 Turbo	2019	NaN	True	['Controle de tração', 'Ar condicionado', 'Cen...	67664.86	0.0
Benni Mini	Motor V8	2019	NaN	True	['Sensor crepuscular', 'Câmbio automático', 'C...	126247.84	0.0
Uno	Motor Diesel V6	2019	NaN	True	['Central multimídia', 'Sensor crepuscular', '...	128852.21	0.0
Santa Fe	Motor 3.0 32v	2019	NaN	True	['Travas elétricas', 'Ar condicionado', '4 X 4...	129415.33	0.0
XC60	Motor 4.0 Turbo	2019	NaN	True	['Painel digital', 'Piloto automático', 'Centr...	77675.79	0.0

61 rows x 7 columns

O NaN foi resultante do carro ser Zero Km. Assim, podemos preencher com zero os valores faltantes da coluna Quilometragem. Faremos da seguinte forma:

```
dataset.fillna(0, inplace = True) #inplace = True para fazer a substituição na base original
```

```
>> dataset.query("Zero_km == True")
```

	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor	Km_media
Nome							
DS5	Motor 2.4 Turbo	2019	0.0	True	['Travas elétricas', '4 X 4', 'Vidros elétrico...	124549.07	0.0
A5	Motor 4.0 Turbo	2019	0.0	True	['Câmbio automático', 'Câmera de estacionamento...	56445.20	0.0
J5	Motor V6	2019	0.0	True	['Sensor crepuscular', 'Painel digital', 'Roda...	53183.38	0.0
A3	Motor 1.0 8v	2019	0.0	True	['4 X 4', 'Piloto automático', 'Central multim...	88552.39	0.0
Série 1 M	Motor V8	2019	0.0	True	['Controle de estabilidade', 'Central multimíd...	94564.40	0.0
...
Lamborghini Reventón	Motor 4.0 Turbo	2019	0.0	True	['Controle de tração', 'Ar condicionado', 'Cen...	67664.86	0.0
Benni Mini	Motor V8	2019	0.0	True	['Sensor crepuscular', 'Câmbio automático', 'C...	126247.84	0.0
Uno	Motor Diesel V6	2019	0.0	True	['Central multimídia', 'Sensor crepuscular', '...	128852.21	0.0
Santa Fe	Motor 3.0 32v	2019	0.0	True	['Travas elétricas', 'Ar condicionado', '4 X 4...	129415.33	0.0
XC60	Motor 4.0 Turbo	2019	0.0	True	['Painel digital', 'Piloto automático', 'Centr...	77675.79	0.0

61 rows x 7 columns

Poderíamos também retirar os valores de NaN na coluna 'Quilometragem' pela famosa função `dropna()`. Segue seu código abaixo:

```
dataset.dropna(subset = ['Quilometragem'], inplace = True)
```

	Nome	Motor	Ano	Quilometragem	Zero_km	Acessórios	Valor
0	Jetta Variant	Motor 4.0 Turbo	2003	44410.0	False	['Rodas de liga', 'Travas elétricas', 'Piloto ...	88078.64
1	Passat	Motor Diesel	1991	5712.0	False	['Central multimídia', 'Teto panorâmico', 'Fre...	106161.94
2	Crossfox	Motor Diesel V8	1990	37123.0	False	['Piloto automático', 'Controle de estabilidade...	72832.16
4	Aston Martin DB4	Motor 2.4 Turbo	2006	25757.0	False	['Rodas de liga', '4 X 4', 'Central multimídia...	92612.10
5	Palio Weekend	Motor 1.8 16v	2012	10728.0	False	['Sensor de estacionamento', 'Teto panorâmico'...	97497.73
...
253	Phantom 2013	Motor V8	2014	27505.0	False	['Controle de estabilidade', 'Piloto automátic...	51759.58
254	Cadillac Ciel concept	Motor V8	1991	29981.0	False	['Bancos de couro', 'Painel digital', 'Sensor ...	51667.06
255	Classe GLK	Motor 5.0 V8 Bi-Turbo	2002	52637.0	False	['Rodas de liga', 'Controle de tração', 'Câmbi...	68934.03
256	Aston Martin DB5	Motor Diesel	1996	7685.0	False	['Ar condicionado', '4 X 4', 'Câmbio automátic...	122110.90
257	Macan	Motor Diesel V6	1992	50188.0	False	['Central multimídia', 'Teto panorâmico', 'Vid...	90381.47

197 rows x 7 columns