Dataset Sintético - BMW Sales

por Júlio César Estevam

Neste estudo, será utilizada a base **'BMW Worldwide Sales Records (2010–2024)'** que está disponível no website **Kaggle**. O intuito deste documento é ensinar como aumentar a quantidade de linhas de qualquer base de dados, mantendo as proporções iniciais.

Link: **https://www.kaggle.com/datasets/ahmadrazakashif/bmw-worldwide-sales-records-20102024**

Leitura do dataset e contabilização de linhas

```
import pandas as pd #importa a biblioteca pandas
import numpy as np #importa a biblioteca numpy
arquivo = pd.read_csv('D:/BMWsalesdata.csv') # lê o arquivo e atribui à uma variável
df = pd.DataFrame(arquivo) # transforma em um DataFrame Pandas
print(len(df)) # conta a quantidade de linhas
```

O *dataset* possuí 50 mil linhas. É o suficiente para um *dataset* de treino, mas é comum que bases de dados encontradas em sites como o **Kaggle** e o **Makeover Monday** possuam até menos que dez mil linhas, por isso a importância deste documento.

Entendimento das colunas

```
df.dtypes #traz o nome das colunas e seu tipo
```

```
Model                object
Year                  int64
Region               object
Color                object
Fuel_Type            object
Transmission         object
Engine_Size_L       float64
Mileage_KM            int64
Price_USD             int64
Sales_Volume          int64
Sales_Classification object
```

Antes da geração do *dataset*, é importante entender os tipos das colunas e mudar as colunas classificadas como '*object*'.

A classificação '*object*' é um tipo padrão da biblioteca *Pandas*, que pode causar problemas para funções específicas, por isso é importante atribuir o tipo correto.

Dataset Sintético - BMW Sales

por Júlio César Estevam

```
df['Sales_Classification'].unique()
```

Para todas as colunas com o tipo '*object*', usei o comando *unique()* para entender se era ideal a transformação para o tipo '*categorical*' ou '*string*'.

O tipo '*categorical*' não deve ser utilizado em colunas com um grande número de valores exclusivos, pois afeta negativamente a performance. A verificação também poderia ser feita usando a função *nunique()* que traz a quantidade de valores únicos.

```
df['Model'] = df['Model'].astype('category') #transforma a coluna em categórica
df['Region'] = df['Region'].astype('category')
df['Color'] = df['Color'].astype('category')
df['Fuel_Type'] = df['Fuel_Type'].astype('category')
df['Transmission'] = df['Transmission'].astype('category')
df['Sales_Classification'] = df['Sales_Classification'].astype('category')
df.dtypes
```

```
Model                category
Year                    int64
Region               category
Color                category
Fuel_Type            category
Transmission         category
Engine_Size_L         float64
Mileage_KM              int64
Price_USD               int64
Sales_Volume            int64
Sales_Classification category
```

Todas as colunas com o tipo '*object*' foram alteradas.

Em outras soluções, é possível encontrar loops *for* que verificam coluna por coluna a quantidade de valores únicos e transformam em '*categorical*' apenas colunas abaixo do parâmetro escolhido.

Para um *DataFrame* com muitas colunas, é uma solução mais eficiente.

Dataset Sintético - BMW Sales

por Júlio César Estevam

Gerando o dataset sem o parâmetro *p*

```
colunas = df.columns.tolist() # adiciona todos os nomes de coluna em uma lista
for i in colunas:
    if isinstance(df[i].dtype, pd.CategoricalDtype): # verifica se a coluna é categórica
        print(df[i].unique()) # traz as categorias
    else:
        print(df[i].name, min(df[i]),max(df[i])) # traz o valor mínimo e máximo
```

Neste código é verificado de forma manual quais são as categorias das colunas do tipo *'categorical'* e quais são os valores mínimos e máximos das colunas numéricas. Esse passo é importante para mapear como cada coluna se comporta.

```
n_samples = 10_000_000 # quantas linhas terão no dataset
data = { # cria o dataset sintético
    'Model': np.random.choice(['3 Series', '5 Series', '7 Series','X1','X3','X5','M3','M5','i3','i8'], n_samples),
    'Year':np.random.randint(2010,2024,n_samples),
    'Region': np.random.choice(['Asia', 'Middle East', 'North America','Europe','Africa','South America'],
n_samples),
    'Color': np.random.choice(['Red', 'Silver', 'Black', 'White', 'Grey','Blue'], n_samples),
    'Fuel_Type': np.random.choice(['Diesel', 'Electric', 'Petrol', 'Hybrid'], n_samples),
    'Sales_Classification': np.random.choice(['High','Low'], n_samples),
    'Transmission': np.random.choice(['Manual', 'Automatic'], n_samples),
    'Engine_Size_L': np.random.uniform(1.5, 5.0, n_samples),
    'Mileage_KM': np.random.randint(3, 199996, n_samples),
    'Price_USD': np.random.randint(30000, 119998, n_samples),
    'Sales_Volume': np.random.randint(100, 9999, n_samples)}
data = pd.DataFrame(data) # transforma em um DataFrame
print(data.head()) # imprime as cinco primeiras linhas
```

Dataset Sintético - BMW Sales

por Júlio César Estevam

	Model	Year	Region	Color	Fuel_Type	Sales_Classification	Transmission	Engine_Size_L	Mileage_KM	Price_USD	Sales_Volume
0	i8	2011	Africa	Grey	Diesel	Low	Manual	2.990809	33193	68183	7381
1	X3	2017	South America	Blue	Diesel	High	Manual	2.039848	107600	62764	5487
2	X3	2022	Asia	Blue	Electric	Low	Manual	3.644865	154786	66131	686
3	X3	2011	Europe	Grey	Hybrid	High	Automatic	3.014640	71105	112755	851
4	7 Series	2011	Europe	White	Diesel	Low	Manual	1.875340	128660	81330	743

O *dataset* das vendas de **BMW** agora possuí 10 milhões de linhas, mantendo valores similares com o *dataset* original.

O problema desta solução são as funções *np.random*. Elas, por padrão, obedecem a probabilidade clássica e geram a mesma probabilidade pra cada valor disponível.

```
print(round(data['Transmission'].value_counts() * 100 /len(data['Transmission']),2))
```

```
Transmission
Automatic    50.01
Manual       49.99
Name: count, dtype: float64
```

Neste exemplo, como esta coluna tem dois valores possíveis, cada um deles aparece em 50 % das vezes. Para colunas com três valores, seria 33,33 % e assim por diante.

Agora, imagine que deseja incrementar um *dataset* de vendas de videogame por década e cada década seja uma categoria de **'1900 - 1910'** até **'2020 - 2030'**. Não faz sentido dividir igualmente o valor de linhas para cada categoria. O *dataset* perderia as tendências originais e traria métricas distintas.

Ao expandir o número de linhas de um *dataset*, é inevitável que certos valores irão mudar e indicadores terão valores diferentes, mas para minimizar isso, sugiro a seguinte solução:

Dataset Sintético - BMW Sales

por Júlio César Estevam

Gerando o dataset com o parâmetro *p*

```
colunas = df.columns.tolist() # traz os nomes das colunas em uma lista
lista = []
comp = len(df) # quantidade de linhas do dataframe
dado = {}
n_samples = 10_000_000 # quantas linhas desejamos no novo dataset
for i in colunas: # itera entre as colunas
    if isinstance(df[i].dtype, pd.CategoricalDtype): # verifica se é categórica
        chaves_valores = df[i].value_counts() # faz a contagem por categoria
        chaves = []
        valores = []
        cont = 0
        for j,h in enumerate(chaves_valores):
            chaves.append(chaves_valores.index[j]) # adiciona as categorias em uma lista
            h = round(h / comp,5) # calcula a frequência em %
            if chaves_valores.index[j] != chaves_valores.index[-1]:
                cont += h # para evitar problemas com arredondamento, garante que a soma       de p será 1
            else:
                h = 1 - cont
            valores.append(h) # adiciona as frequências em uma lista
        dado.update({
        df[i].name :
        np.random.choice(chaves,n_samples,p=valores)} # cria uma coluna no novo dataset
    )
    else: # cria uma coluna no novo dataset
        dado.update({
        df[i].name :
            np.random.randint(min(df[i]),max(df[i]),n_samples)})
```

Dataset Sintético - BMW Sales

por Júlio César Estevam

	Year	Engine_Size_L	Mileage_KM	Price_USD	Sales_Volume	Model	Region	Color	Fuel_Type	Transmission	Sales_Classification
0	2016	4	55759	51597	4620	3 Series	Middle East	Red	Hybrid	Manual	Low
1	2012	2	179572	40134	1763	X1	North America	Red	Diesel	Automatic	Low
2	2015	3	16484	59050	9983	3 Series	Asia	Black	Diesel	Manual	Low
3	2019	3	150818	72417	971	X1	South America	White	Electric	Manual	Low
4	2014	1	10180	55221	5281	M3	Africa	Grey	Hybrid	Manual	High
...	...	...	...	...	...	...	...	...	...	...	...
9999995	2021	1	132942	31826	1277	M3	Middle East	Blue	Diesel	Automatic	Low
9999996	2023	2	123384	56481	5147	X1	Middle East	Blue	Petrol	Manual	Low
9999997	2013	3	124467	58458	2696	5 Series	Africa	Black	Electric	Automatic	High
9999998	2020	4	193296	96072	662	X1	Europe	Blue	Hybrid	Automatic	Low
9999999	2016	3	148648	54229	3019	X1	Asia	Blue	Hybrid	Automatic	Low

Primeiramente, os nomes das colunas são transformadas em uma lista para que possam ser acessados. Depois, um for loop itera sobre a lista para tratar coluna por coluna e verificar, inicialmente se o tipo é 'categorical' ou não.

Para as categóricas, a função *value_counts()* é utilizada para trazer quantas vezes cada categoria aparece no *dataset* original. O *index* desse resultado será usado como o nome da coluna no novo *dataset* e os valores serão divididos pelo total de linhas para gerar a frequência em %.

Depois, garantimos que a soma de *p* será 1 truncando a última categoria como um menos a soma das outras categorias. Logo depois, criamos a coluna.

Se a coluna não for categórica, será gerada uma coluna com o nome da coluna e N valores entre o mínimo e o máximo valor daquela coluna. Para as colunas numéricas, não houve diferença na geração, pois a função *randint* não permite a parametrização de *p*.

Além de alterar as proporções iniciais, a solução prévia, com a necessidade de preenchimento manual, é inviável para bases longas, por isso recomendo somente o uso da solução com a parametrização de *p*.

*Neste tratamento não há tratamento para colunas com tipo *'string'*. Colunas deste tipo apareceriam se possuíssem um número grande de valores únicos impossibilitando a conversão para *'categorical'*. Colunas como nome, município e CNPJ perderiam o sentido numa expansão de *dataset*, por isso optei por não trazê-las para o código.