RACIOCÍNIO BASEADO EM CASOS EM PREVISÃO DE PREÇOS DE CARROS

5/3/2024

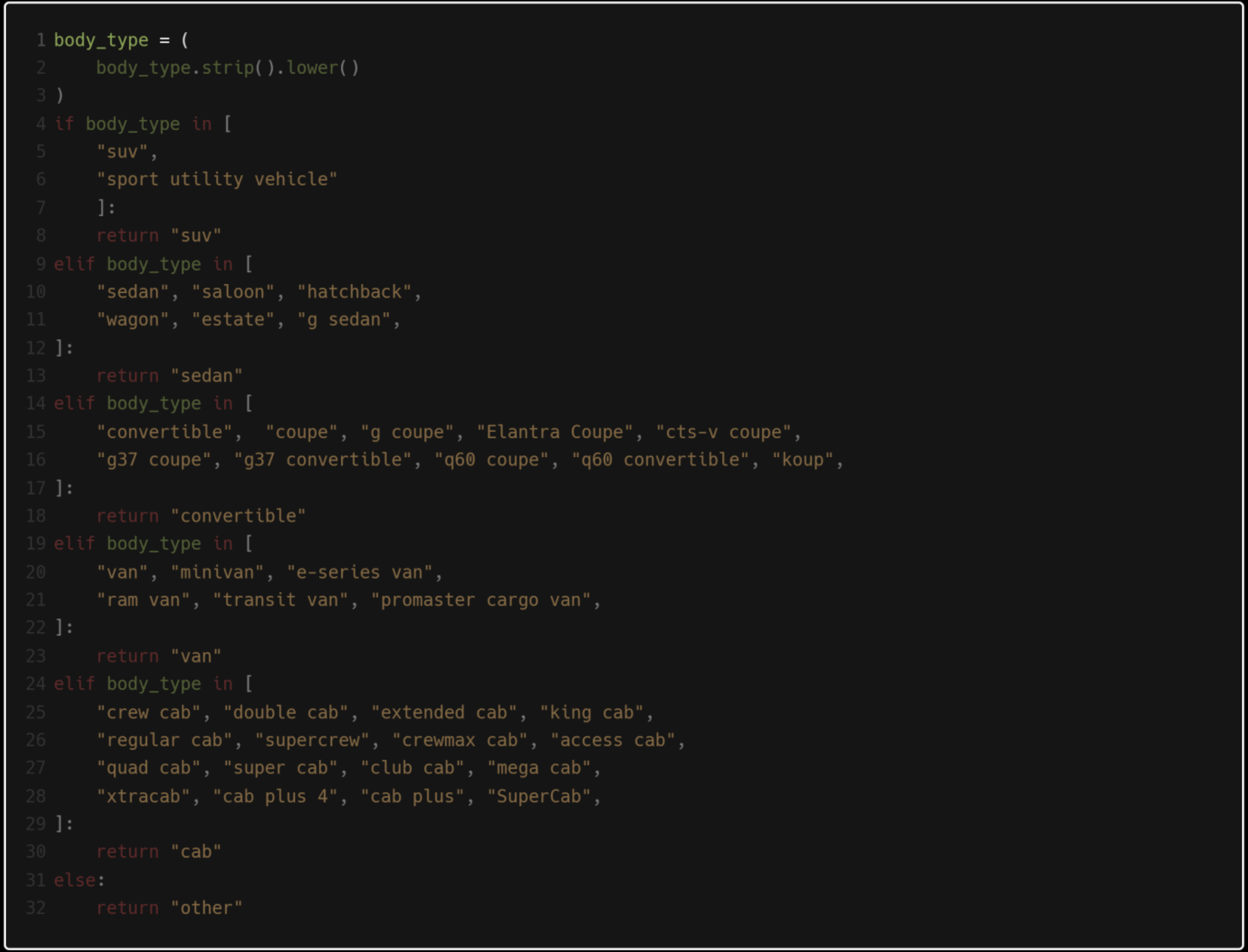
Eduardo Savian, Marcos Augusto Fehlauer Pereira

Escola Politécnica - UNIVALI

INTRODUÇÃO

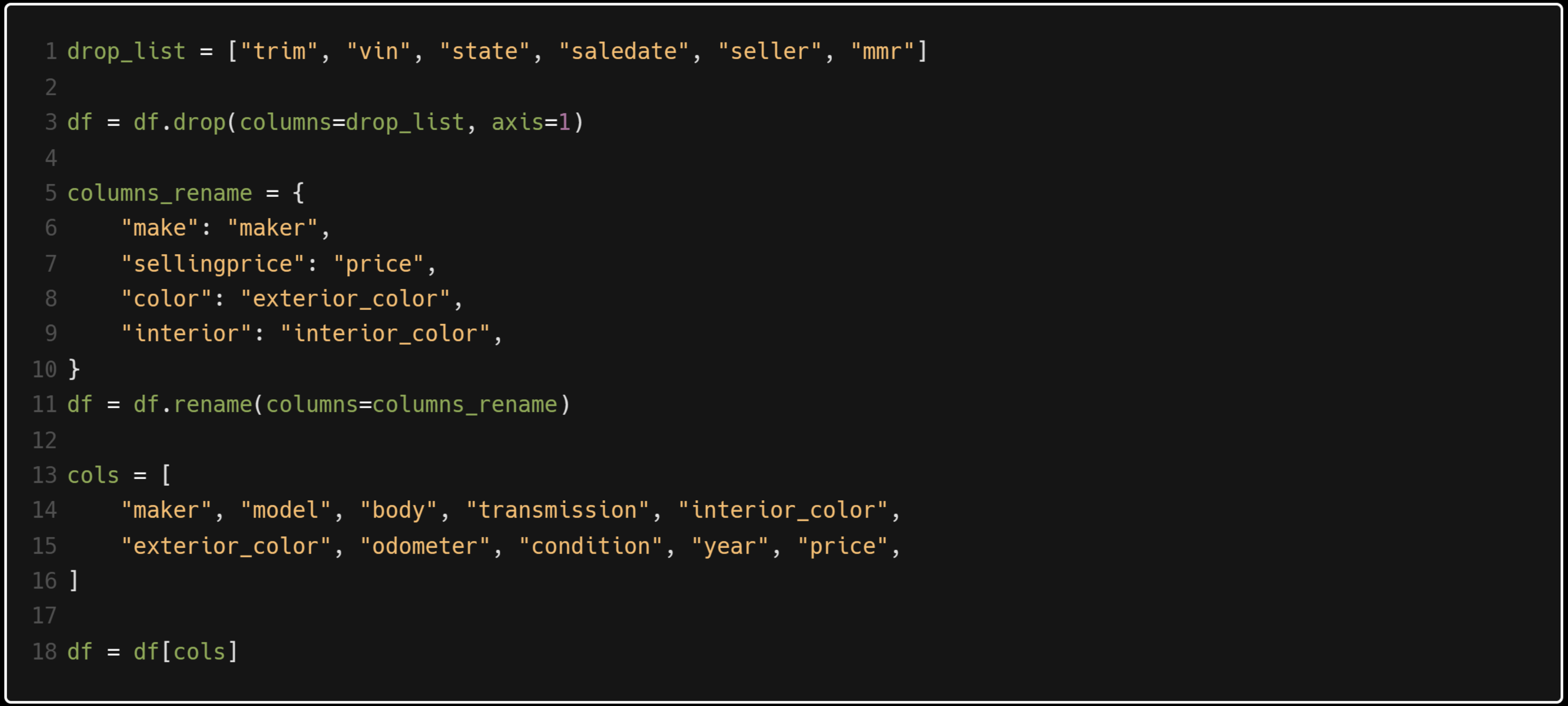
- A abordagem do raciocínio baseado em casos (RBC) é utilizada para resolver novos problemas adaptando soluções previamente aplicadas em problemas anteriores.
- O método de RBC foi aplicado em previsão de custo de um carro com base nos parâmetros de marca, modelo, tipo de carroceria, cores interna e externa, odômetro, condição e ano, cada um com seus respectivos pesos.
- A base de dados original contém 550.298 itens únicos, que precisam passar por um processo de limpeza.
- Para otimizar a execução do processo, foi selecionado um subconjunto que contêm 10% da base original, para criar uma versão menor e mais eficiente que será utilizada pelo aplicativo.

AGRUPANDO DADOS EM CATEGORIAS GERAIS



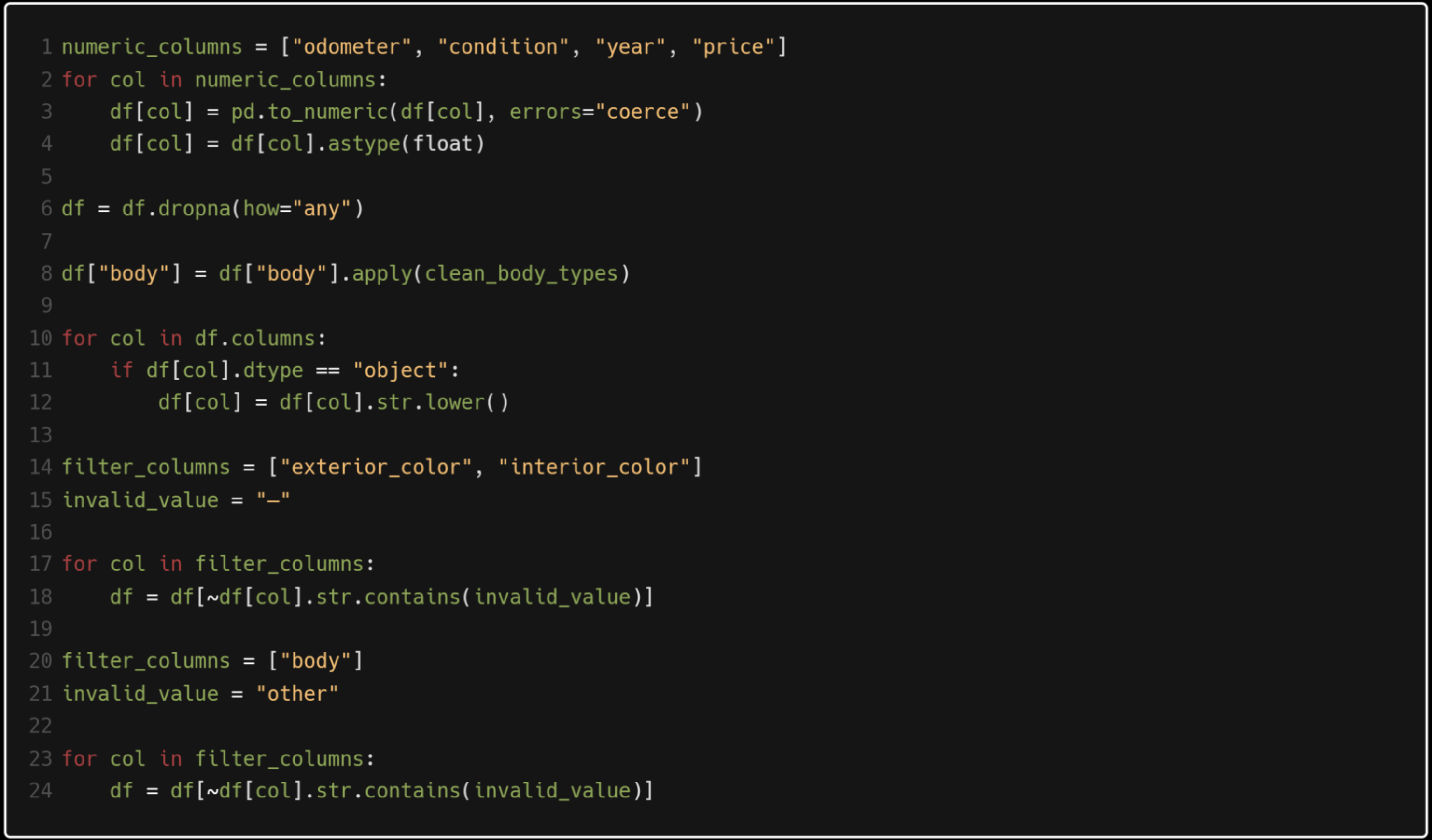
```
body_type = (
    body_type.strip().lower()
)
if body_type in [
    "suv",
    "sport utility vehicle"
    ]:
    return "suv"
elif body_type in [
    "sedan", "saloon", "hatchback",
    "wagon", "estate", "g sedan",
]:
    return "sedan"
elif body_type in [
    "convertible",  "coupe", "g coupe", "Elantra Coupe", "cts-v coupe",
    "g37 coupe", "g37 convertible", "q60 coupe", "q60 convertible", "koup",
]:
    return "convertible"
elif body_type in [
    "van", "minivan", "e-series van",
    "ram van", "transit van", "promaster cargo van",
]:
    return "van"
elif body_type in [
    "crew cab", "double cab", "extended cab", "king cab",
    "regular cab", "supercrew", "crewmax cab", "access cab",
    "quad cab", "super cab", "club cab", "mega cab",
    "xtracab", "cab plus 4", "cab plus", "SuperCab",
]:
    return "cab"
else:
    return "other"
```

ORGANIZANDO O *DATAFRAME*



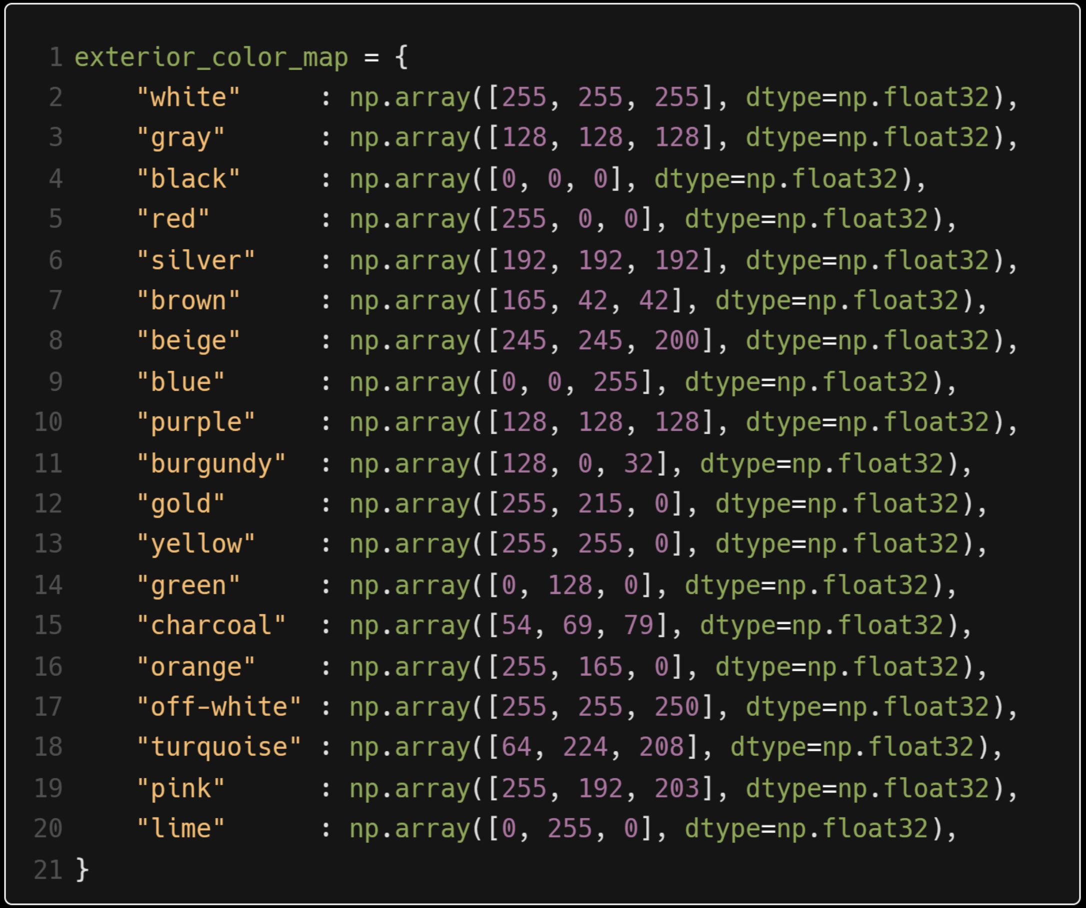
```python
drop_list = ["trim", "vin", "state", "saledate", "seller", "mmr"]

df = df.drop(columns=drop_list, axis=1)

columns_rename = {
    "make": "maker",
    "sellingprice": "price",
    "color": "exterior_color",
    "interior": "interior_color",
}
df = df.rename(columns=columns_rename)

cols = [
    "maker", "model", "body", "transmission", "interior_color",
    "exterior_color", "odometer", "condition", "year", "price",
]

df = df[cols]
```

LIMPANDO O *DATAFRAME*



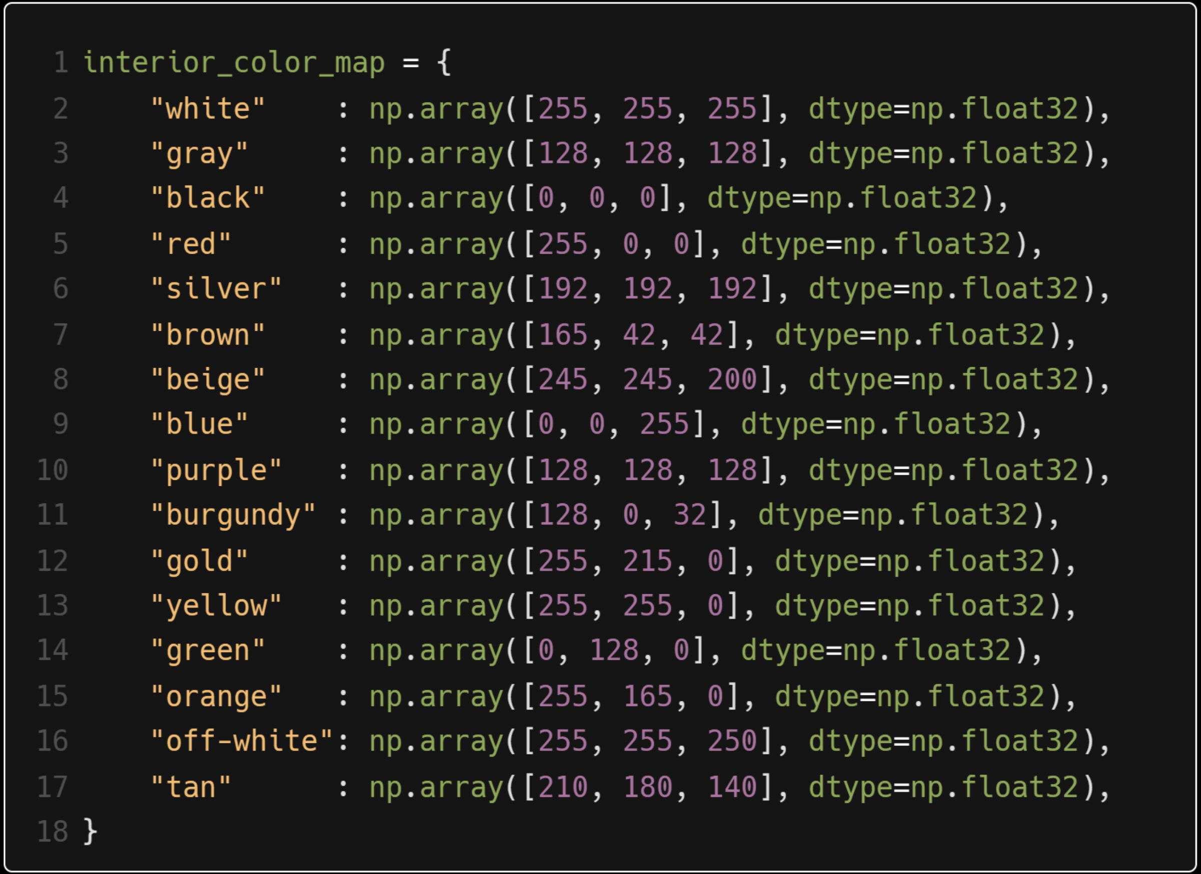
```python
numeric_columns = ["odometer", "condition", "year", "price"]
for col in numeric_columns:
    df[col] = pd.to_numeric(df[col], errors="coerce")
    df[col] = df[col].astype(float)

df = df.dropna(how="any")

df["body"] = df["body"].apply(clean_body_types)

for col in df.columns:
    if df[col].dtype == "object":
        df[col] = df[col].str.lower()

filter_columns = ["exterior_color", "interior_color"]
invalid_value = "—"

for col in filter_columns:
    df = df[~df[col].str.contains(invalid_value)]

filter_columns = ["body"]
invalid_value = "other"

for col in filter_columns:
    df = df[~df[col].str.contains(invalid_value)]
```

SIMILARIDADE DE CORES - TABELA



```python
exterior_color_map = {
    "white"     : np.array([255, 255, 255], dtype=np.float32),
    "gray"      : np.array([128, 128, 128], dtype=np.float32),
    "black"     : np.array([0, 0, 0], dtype=np.float32),
    "red"       : np.array([255, 0, 0], dtype=np.float32),
    "silver"    : np.array([192, 192, 192], dtype=np.float32),
    "brown"     : np.array([165, 42, 42], dtype=np.float32),
    "beige"     : np.array([245, 245, 200], dtype=np.float32),
    "blue"      : np.array([0, 0, 255], dtype=np.float32),
    "purple"    : np.array([128, 128, 128], dtype=np.float32),
    "burgundy"  : np.array([128, 0, 32], dtype=np.float32),
    "gold"      : np.array([255, 215, 0], dtype=np.float32),
    "yellow"    : np.array([255, 255, 0], dtype=np.float32),
    "green"     : np.array([0, 128, 0], dtype=np.float32),
    "charcoal"  : np.array([54, 69, 79], dtype=np.float32),
    "orange"    : np.array([255, 165, 0], dtype=np.float32),
    "off-white" : np.array([255, 255, 250], dtype=np.float32),
    "turquoise" : np.array([64, 224, 208], dtype=np.float32),
    "pink"      : np.array([255, 192, 203], dtype=np.float32),
    "lime"      : np.array([0, 255, 0], dtype=np.float32),
}
```

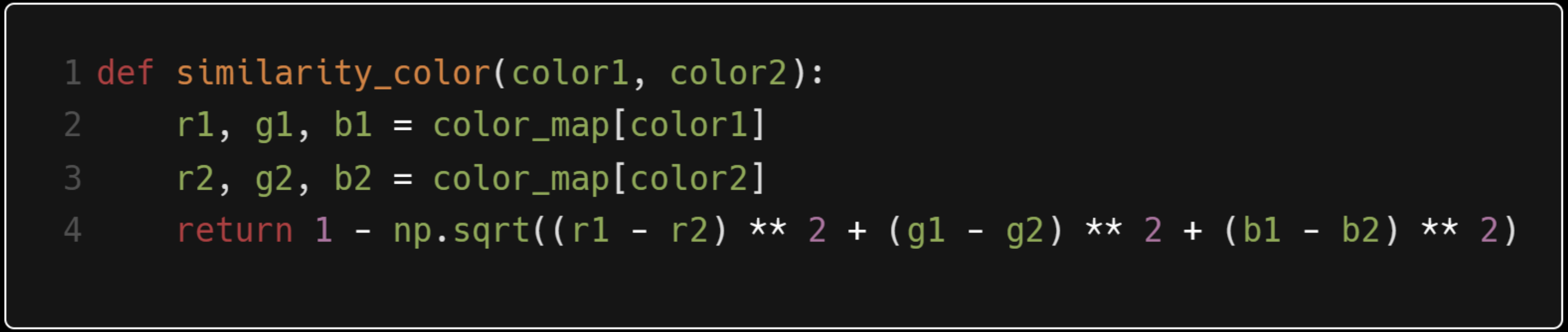
SIMILARIDADE DE CORES - TABELA



```python
interior_color_map = {
    "white"    : np.array([255, 255, 255], dtype=np.float32),
    "gray"     : np.array([128, 128, 128], dtype=np.float32),
    "black"    : np.array([0, 0, 0], dtype=np.float32),
    "red"      : np.array([255, 0, 0], dtype=np.float32),
    "silver"   : np.array([192, 192, 192], dtype=np.float32),
    "brown"    : np.array([165, 42, 42], dtype=np.float32),
    "beige"    : np.array([245, 245, 200], dtype=np.float32),
    "blue"     : np.array([0, 0, 255], dtype=np.float32),
    "purple"   : np.array([128, 128, 128], dtype=np.float32),
    "burgundy" : np.array([128, 0, 32], dtype=np.float32),
    "gold"     : np.array([255, 215, 0], dtype=np.float32),
    "yellow"   : np.array([255, 255, 0], dtype=np.float32),
    "green"    : np.array([0, 128, 0], dtype=np.float32),
    "orange"   : np.array([255, 165, 0], dtype=np.float32),
    "off-white": np.array([255, 255, 250], dtype=np.float32),
    "tan"      : np.array([210, 180, 140], dtype=np.float32),
}
```

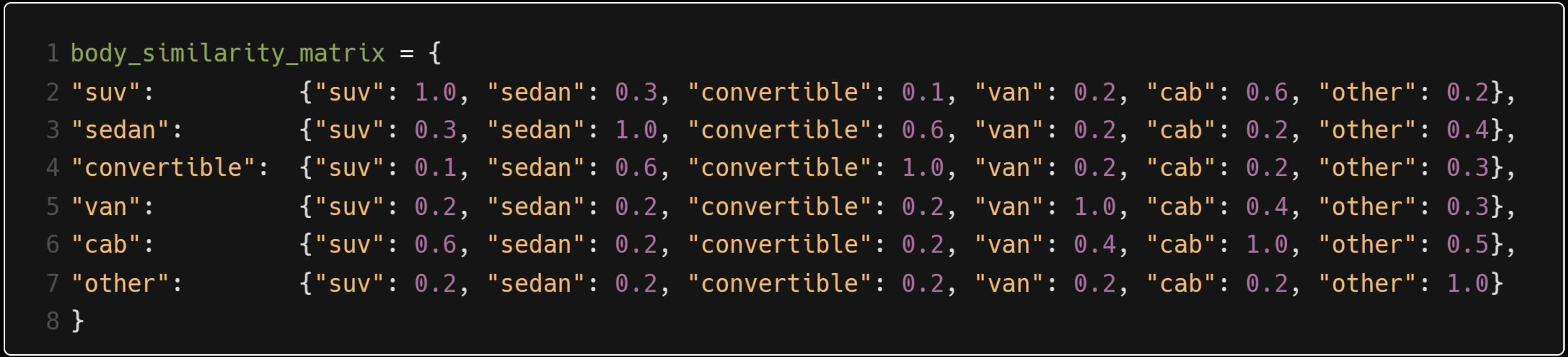
SIMILARIDADE DE CORES

A similaridade de cores é definida pela sua distância euclidiana no espaço RGB.



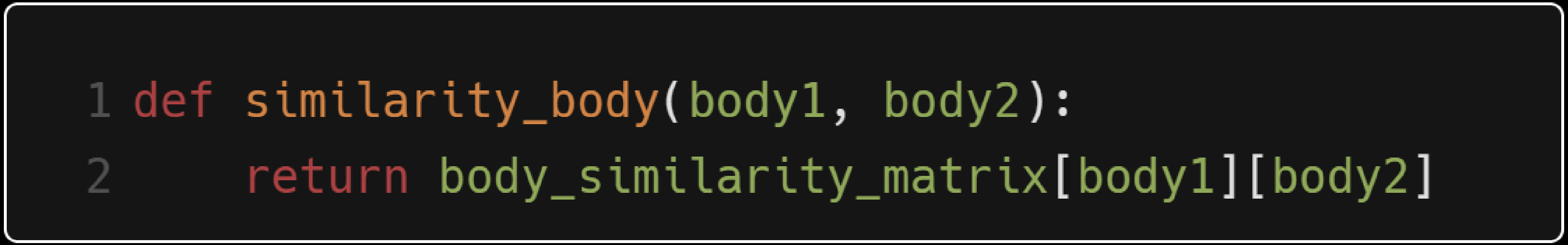
```python
def similarity_color(color1, color2):
    r1, g1, b1 = color_map[color1]
    r2, g2, b2 = color_map[color2]
    return 1 - np.sqrt((r1 - r2) ** 2 + (g1 - g2) ** 2 + (b1 - b2) ** 2)
```

SIMILARIDADE DE CORPOS - TABELA



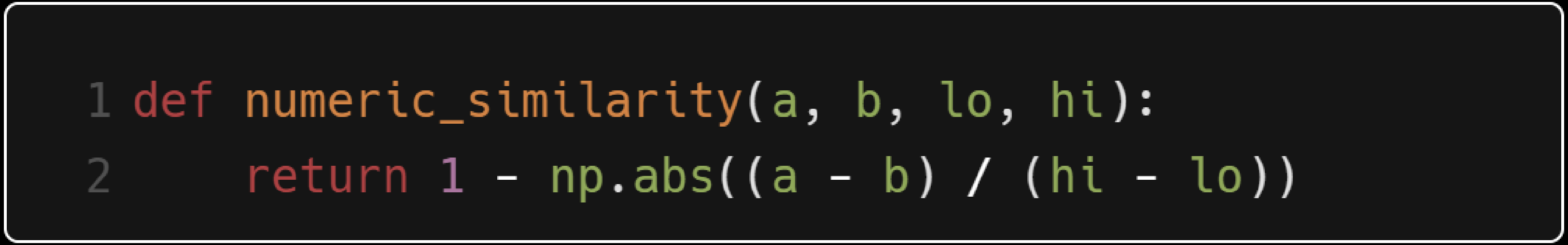
```
body_similarity_matrix = {
"suv":          {"suv": 1.0, "sedan": 0.3, "convertible": 0.1, "van": 0.2, "cab": 0.6, "other": 0.2},
"sedan":        {"suv": 0.3, "sedan": 1.0, "convertible": 0.6, "van": 0.2, "cab": 0.2, "other": 0.4},
"convertible":  {"suv": 0.1, "sedan": 0.6, "convertible": 1.0, "van": 0.2, "cab": 0.2, "other": 0.3},
"van":          {"suv": 0.2, "sedan": 0.2, "convertible": 0.2, "van": 1.0, "cab": 0.4, "other": 0.3},
"cab":          {"suv": 0.6, "sedan": 0.2, "convertible": 0.2, "van": 0.4, "cab": 1.0, "other": 0.5},
"other":        {"suv": 0.2, "sedan": 0.2, "convertible": 0.2, "van": 0.2, "cab": 0.2, "other": 1.0}
}
```

SIMILARIDADE DE CORPOS

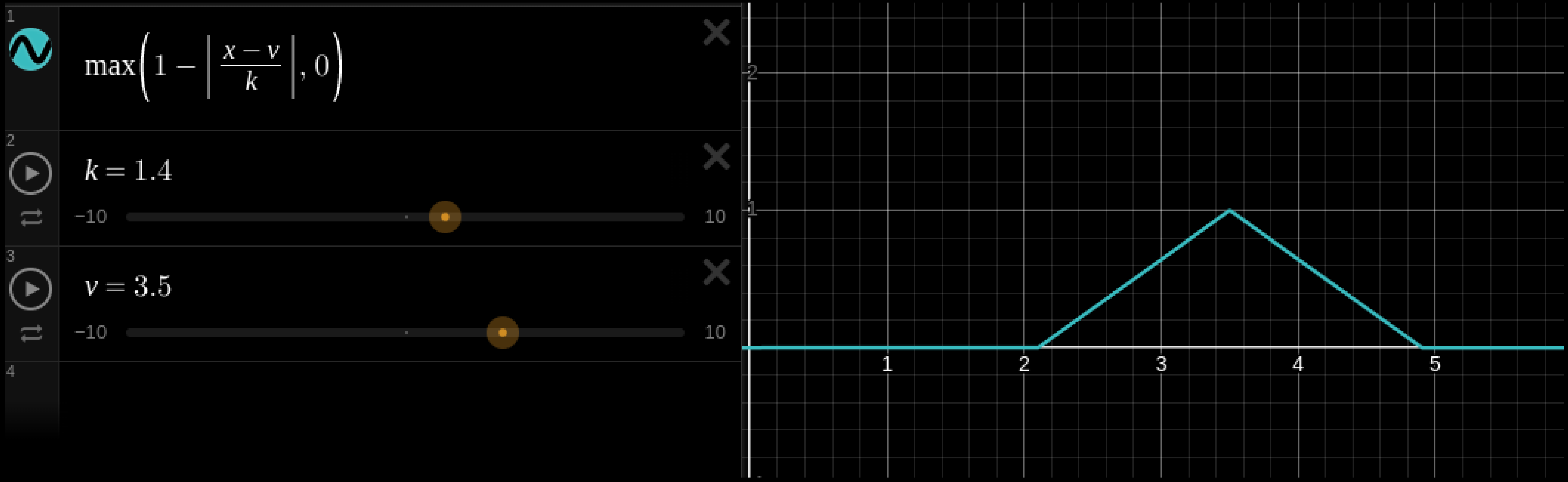


```python
def similarity_body(body1, body2):
    return body_similarity_matrix[body1][body2]
```

SIMILARIDADE NUMÉRICA



```python
def numeric_similarity(a, b, lo, hi):
    return 1 - np.abs((a - b) / (hi - lo))
```



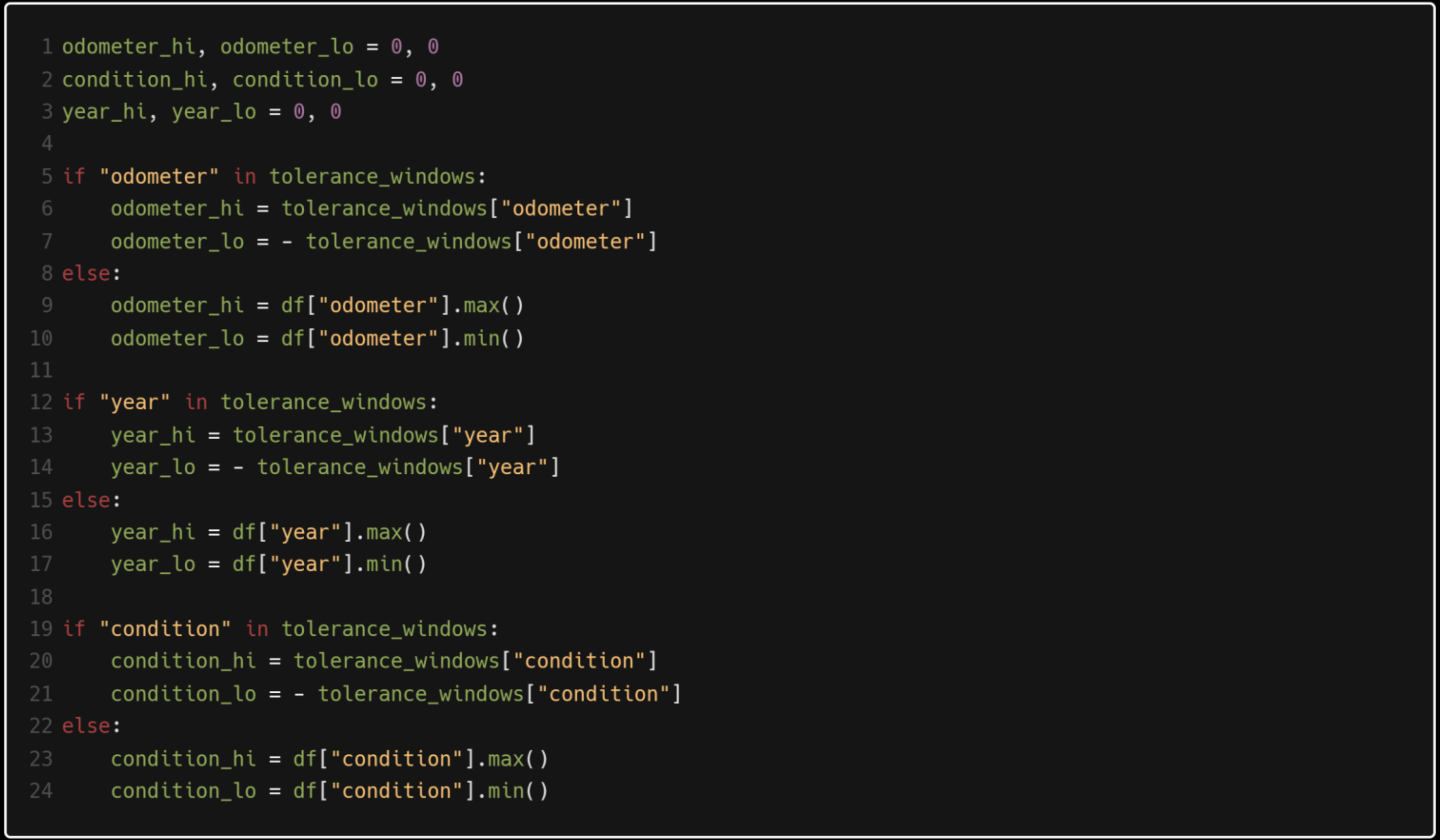
$$\max\left(1-\left|\frac{x-v}{k}\right|,0\right)$$

$k=1.4$

$v=3.5$

SIMILARIDADE NUMÉRICA

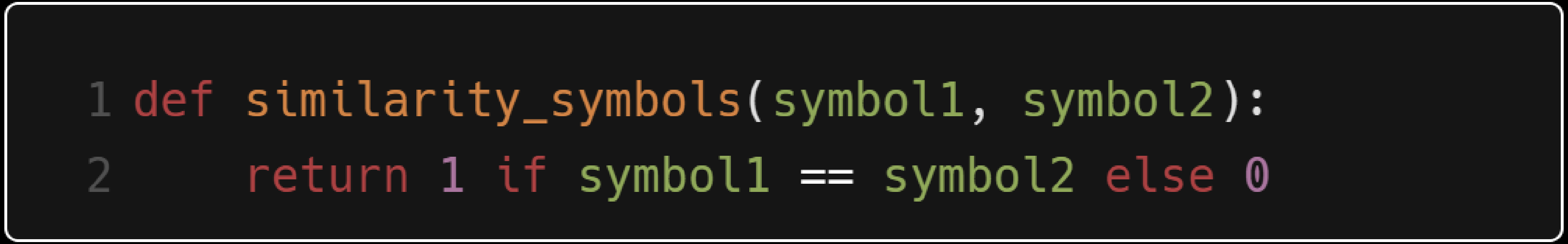
Por padrão a janela de similaridade segue os máximos e mínimos do dataset, o usuário pode opcionalmente sobrescrever essa janela, permitindo um controle mais granular sobre a consulta.



```python
odometer_hi, odometer_lo = 0, 0
condition_hi, condition_lo = 0, 0
year_hi, year_lo = 0, 0

if "odometer" in tolerance_windows:
    odometer_hi = tolerance_windows["odometer"]
    odometer_lo = - tolerance_windows["odometer"]
else:
    odometer_hi = df["odometer"].max()
    odometer_lo = df["odometer"].min()

if "year" in tolerance_windows:
    year_hi = tolerance_windows["year"]
    year_lo = - tolerance_windows["year"]
else:
    year_hi = df["year"].max()
    year_lo = df["year"].min()

if "condition" in tolerance_windows:
    condition_hi = tolerance_windows["condition"]
    condition_lo = - tolerance_windows["condition"]
else:
    condition_hi = df["condition"].max()
    condition_lo = df["condition"].min()
```

SIMILARIDADE SIMBÓLICA

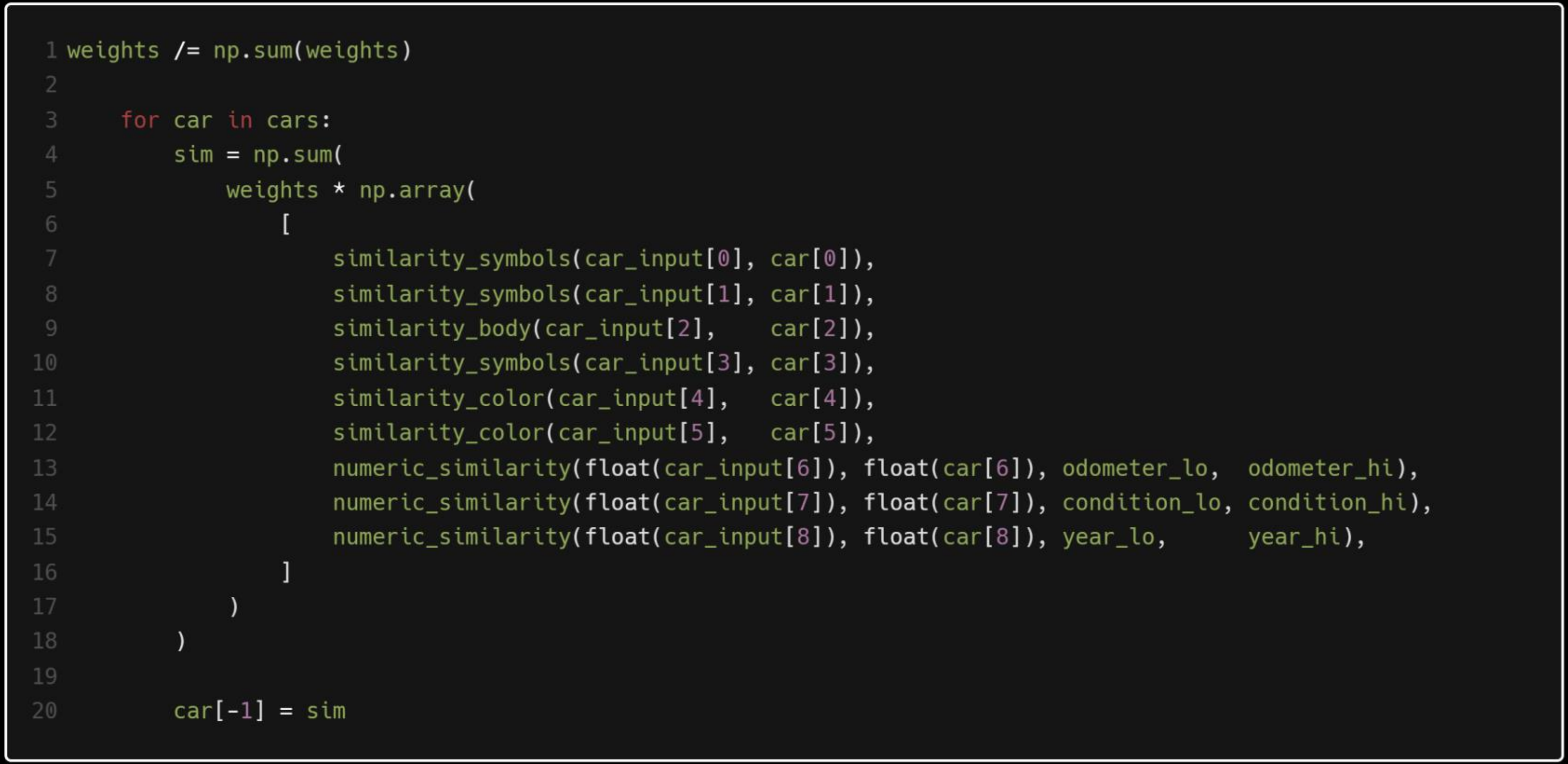
A similaridade entre símbolos é apenas uma identidade pura.



```python
def similarity_symbols(symbol1, symbol2):
    return 1 if symbol1 == symbol2 else 0
```

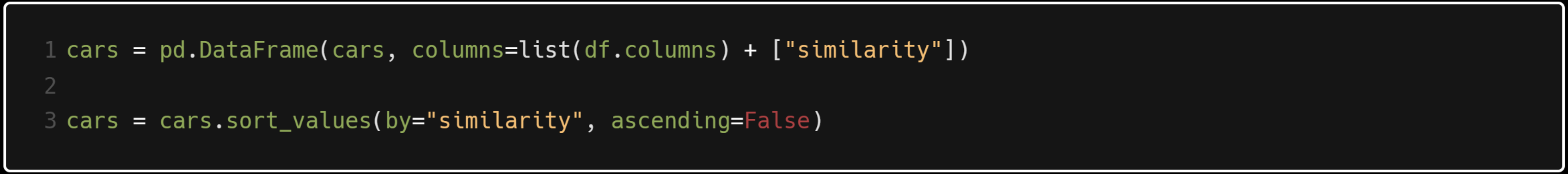
CÁLCULO DA SIMILARIDADE

Cada campo possui seu grau de similaridade calculado e multiplicado pelo vetor normalizado de pesos.



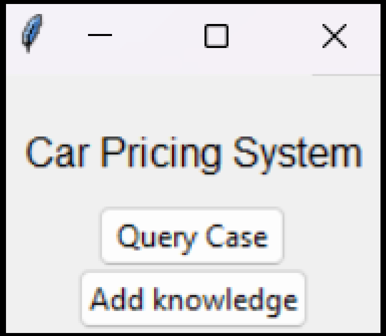
```
weights /= np.sum(weights)

    for car in cars:
        sim = np.sum(
            weights * np.array(
                [
                    similarity_symbols(car_input[0], car[0]),
                    similarity_symbols(car_input[1], car[1]),
                    similarity_body(car_input[2],    car[2]),
                    similarity_symbols(car_input[3], car[3]),
                    similarity_color(car_input[4],   car[4]),
                    similarity_color(car_input[5],   car[5]),
                    numeric_similarity(float(car_input[6]), float(car[6]), odometer_lo,  odometer_hi),
                    numeric_similarity(float(car_input[7]), float(car[7]), condition_lo, condition_hi),
                    numeric_similarity(float(car_input[8]), float(car[8]), year_lo,      year_hi),
                ]
            )
        )

        car[-1] = sim
```

TRANSFORMANDO O *ARRAY* EM *DATAFRAME*

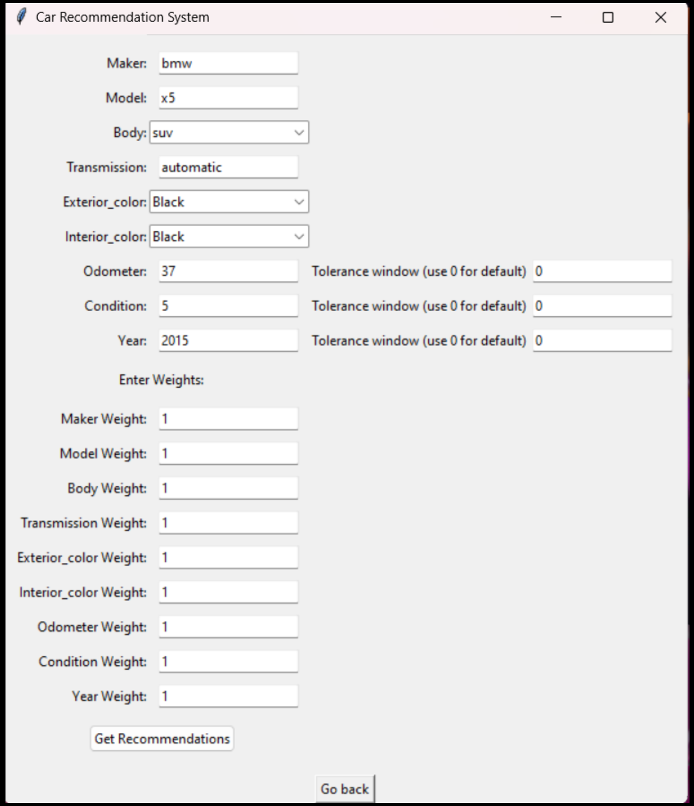


```python
cars = pd.DataFrame(cars, columns=list(df.columns) + ["similarity"])

cars = cars.sort_values(by="similarity", ascending=False)
```

MENU PRINCIPAL


Car Pricing System

PESQUISA DE ITEM



Maker: bmw

Model: x5

Body: suv

Transmission: automatic

Exterior_color: Black

Interior_color: Black

Odometer: 37 Tolerance window (use 0 for default) 0

Condition: 5 Tolerance window (use 0 for default) 0

Year: 2015 Tolerance window (use 0 for default) 0

Enter Weights:

Maker Weight: 1

Model Weight: 1

Body Weight: 1

Transmission Weight: 1

Exterior_color Weight: 1

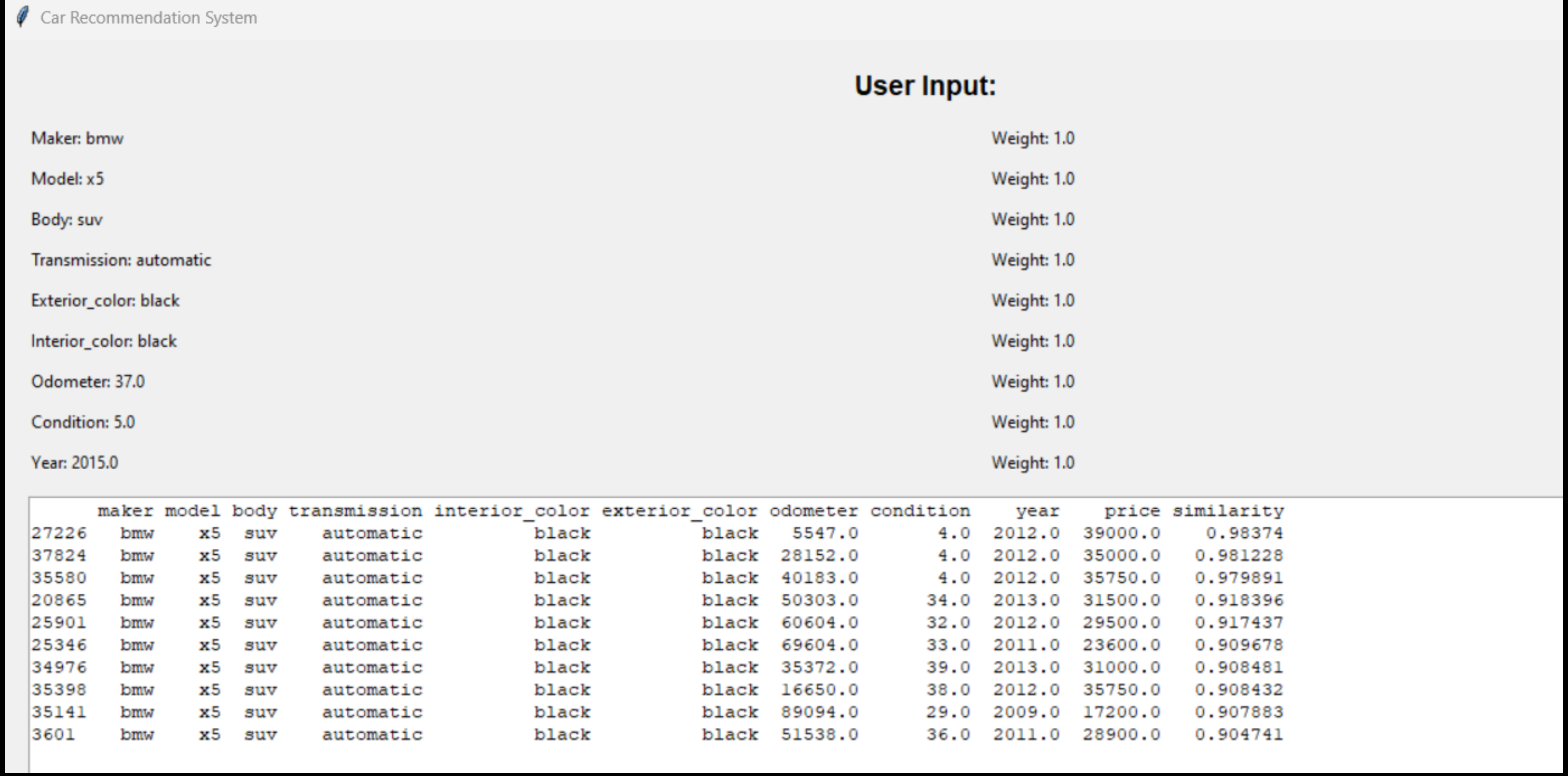
Interior_color Weight: 1

Odometer Weight: 1

Condition Weight: 1

Year Weight: 1

RESULTADO DA PESQUISA



User Input:

Maker: bmw Weight: 1.0

Model: x5 Weight: 1.0

Body: suv Weight: 1.0

Transmission: automatic Weight: 1.0

Exterior_color: black Weight: 1.0

Interior_color: black Weight: 1.0

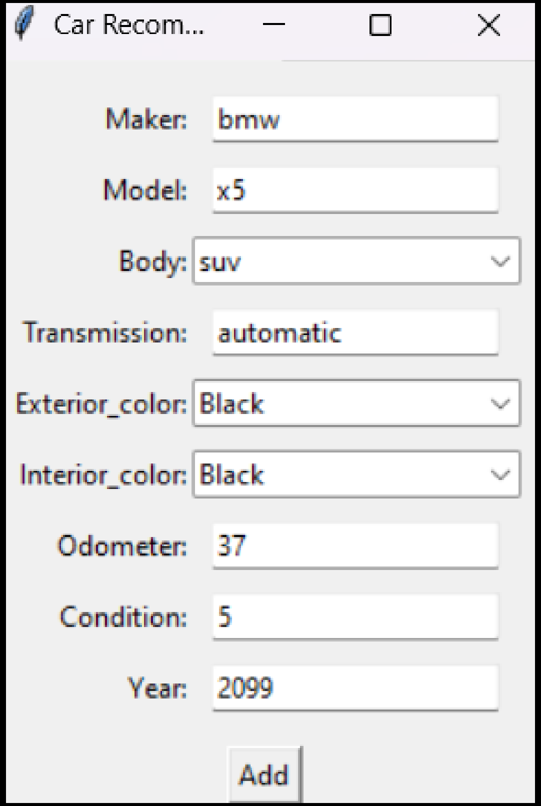
Odometer: 37.0 Weight: 1.0

Condition: 5.0 Weight: 1.0

Year: 2015.0 Weight: 1.0

```
      maker model body transmission interior_color exterior_color odometer condition   year   price similarity
27226   bmw    x5  suv    automatic          black          black   5547.0       4.0 2012.0 39000.0    0.98374
37824   bmw    x5  suv    automatic          black          black  28152.0       4.0 2012.0 35000.0   0.981228
35580   bmw    x5  suv    automatic          black          black  40183.0       4.0 2012.0 35750.0   0.979891
20865   bmw    x5  suv    automatic          black          black  50303.0      34.0 2013.0 31500.0   0.918396
25901   bmw    x5  suv    automatic          black          black  60604.0      32.0 2012.0 29500.0   0.917437
25346   bmw    x5  suv    automatic          black          black  69604.0      33.0 2011.0 23600.0   0.909678
34976   bmw    x5  suv    automatic          black          black  35372.0      39.0 2013.0 31000.0   0.908481
35398   bmw    x5  suv    automatic          black          black  16650.0      38.0 2012.0 35750.0   0.908432
35141   bmw    x5  suv    automatic          black          black  89094.0      29.0 2009.0 17200.0   0.907883
3601    bmw    x5  suv    automatic          black          black  51538.0      36.0 2011.0 28900.0   0.904741
```

ADICIONANDO ITEM NA BASE DE CONHECIMENTO



Car Recom...

Maker: bmw

Model: x5

Body: suv

Transmission: automatic

Exterior_color: Black

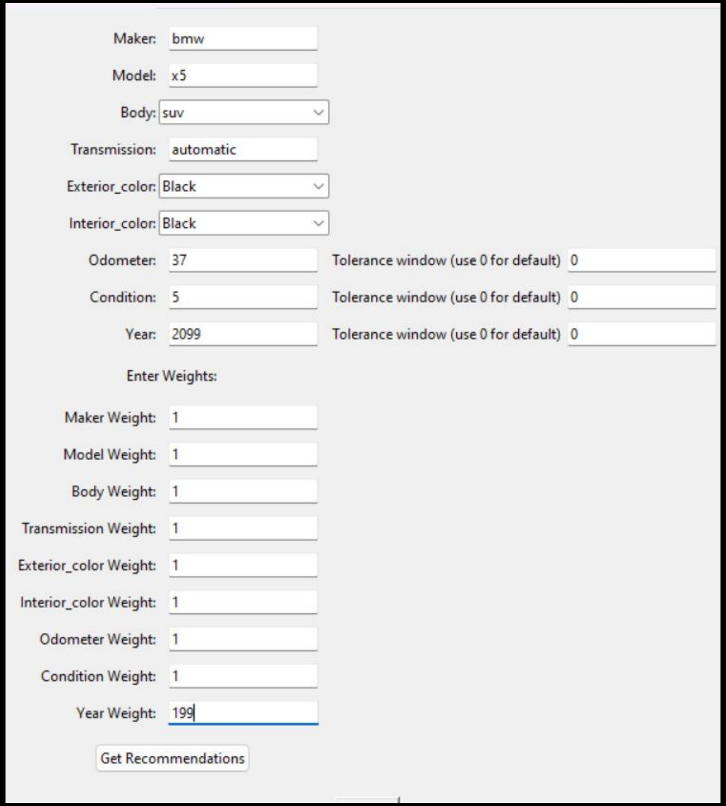
Interior_color: Black

Odometer: 37

Condition: 5

Year: 2099

PESQUISA DO ITEM ADICIONADO



Maker: bmw

Model: x5

Body: suv

Transmission: automatic

Exterior_color: Black

Interior_color: Black

Odometer: 37 Tolerance window (use 0 for default) 0

Condition: 5 Tolerance window (use 0 for default) 0

Year: 2099 Tolerance window (use 0 for default) 0

Enter Weights:

Maker Weight: 1

Model Weight: 1

Body Weight: 1

Transmission Weight: 1

Exterior_color Weight: 1

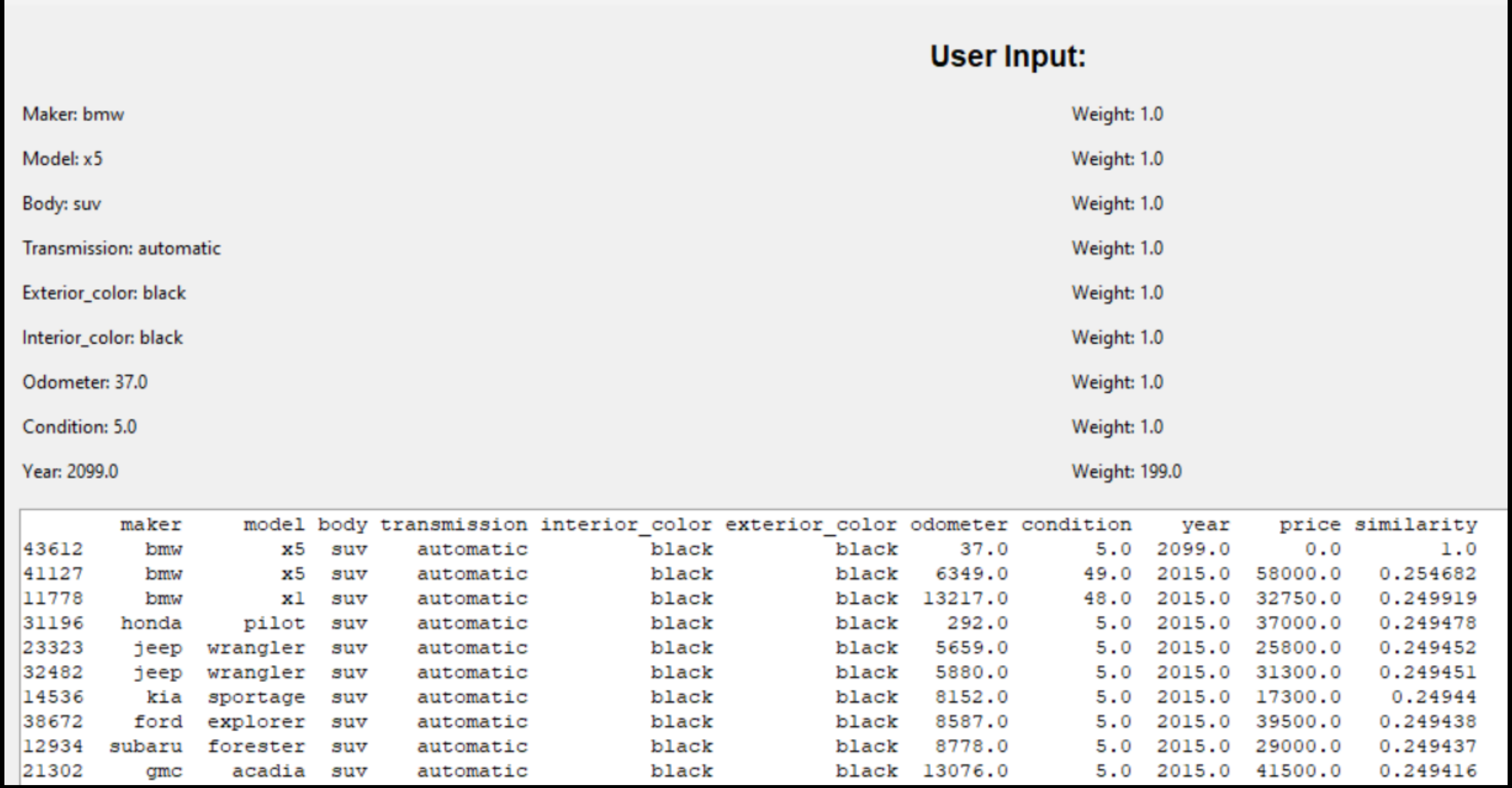
Interior_color Weight: 1

Odometer Weight: 1

Condition Weight: 1

Year Weight: 199

RESULTADO DA PESQUISA



User Input:

Maker: bmw	Weight: 1.0
Model: x5	Weight: 1.0
Body: suv	Weight: 1.0
Transmission: automatic	Weight: 1.0
Exterior_color: black	Weight: 1.0
Interior_color: black	Weight: 1.0
Odometer: 37.0	Weight: 1.0
Condition: 5.0	Weight: 1.0
Year: 2099.0	Weight: 199.0

```
       maker    model body transmission interior_color exterior_color odometer condition   year   price similarity
43612    bmw       x5  suv    automatic          black          black     37.0       5.0 2099.0     0.0        1.0
41127    bmw       x5  suv    automatic          black          black   6349.0      49.0 2015.0 58000.0   0.254682
11778    bmw       x1  suv    automatic          black          black  13217.0      48.0 2015.0 32750.0   0.249919
31196  honda    pilot  suv    automatic          black          black    292.0       5.0 2015.0 37000.0   0.249478
23323   jeep wrangler  suv    automatic          black          black   5659.0       5.0 2015.0 25800.0   0.249452
32482   jeep wrangler  suv    automatic          black          black   5880.0       5.0 2015.0 31300.0   0.249451
14536    kia sportage  suv    automatic          black          black   8152.0       5.0 2015.0 17300.0    0.24944
38672   ford explorer  suv    automatic          black          black   8587.0       5.0 2015.0 39500.0   0.249438
12934 subaru forester  suv    automatic          black          black   8778.0       5.0 2015.0 29000.0   0.249437
21302    gmc   acadia  suv    automatic          black          black  13076.0       5.0 2015.0 41500.0   0.249416
```

REFERÊNCIAS BIBLIOGRÁFICAS

- WIKIPEDIA CONTRIBUTORS. Case-based reasoning. Disponível em: <https://en.wikipedia.org/wiki/Case-based_reasoning>.

- Vehicle Sales Data. Disponível em: <https://www.kaggle.com/datasets/syedanwarafridi/vehicle-sales-data>.

OBRIGADO