1. Análise exploratória dos dados

Importação de bibliotecas

In [93]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [94]:
```
# Bibliotecas de ML
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.preprocessing import LabelEncoder

# Métricas de avalização dos modelos
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

a. Carregue a base de dados media_precos_carros_brasil.csv

In [95]:
```
dados = pd.read_csv('precos_carros_brasil.csv')
```

In [96]:
```
dados.columns
```

Out[96]:
```
Index(['year_of_reference', 'month_of_reference', 'fipe_code',
       'authentication', 'brand', 'model', 'fuel', 'gear', 'engine_size',
       'year_model', 'avg_price_brl'],
      dtype='object')
```

b. Verifique se há valores faltantes nos dados. Caso haja, escolha uma tratativa para resolver o problema de valores faltantes

In [97]:
```
dados.head()
```

Out[97]:

	year_of_reference	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size	year_model	avg_price_brl
0	2021.0	January	004001-0	cfzlctzfwrcp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2002.0	9162.0
1	2021.0	January	004001-0	cdqwxwpw3y2p	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2001.0	8832.0
2	2021.0	January	004001-0	cb1t3xwwj1xp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2000.0	8388.0
3	2021.0	January	004001-0	cb9gct6j65r0	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Alcohol	manual	1	2000.0	8453.0
4	2021.0	January	004003-7	g15wg0gbz1fx	GM - Chevrolet	Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI	Gasoline	manual	1,6	2001.0	12525.0

In [98]:
```
dados.shape
```

Out[98]:
```
(267542, 11)
```

In [99]:
```
dados.isna().any()
```

Out[99]:
```
year_of_reference     True
month_of_reference    True
fipe_code             True
authentication        True
brand                 True
model                 True
fuel                  True
gear                  True
engine_size           True
year_model            True
avg_price_brl         True
dtype: bool
```

In [100…
```
dados.isna().sum()
```

Out[100…
```
year_of_reference     65245
month_of_reference    65245
fipe_code             65245
authentication        65245
brand                 65245
model                 65245
fuel                  65245
gear                  65245
engine_size           65245
year_model            65245
avg_price_brl         65245
dtype: int64
```

In [101…
```
# Apaga os registros onde as linhas estão completamente vazias
dados.dropna(how='all', inplace=True)
```

In [102…
```
dados.isna().sum()
```

Out[102…
```
year_of_reference     0
month_of_reference    0
fipe_code             0
authentication        0
brand                 0
model                 0
fuel                  0
gear                  0
engine_size           0
year_model            0
avg_price_brl         0
dtype: int64
```

c. Verifique se há dados duplicados nos dados

In [103…
```
# Localiza duplicados
dados.duplicated().sum()
```

Out[103…
```
np.int64(2)
```

In [104…
```
# Imprime dados duplicados
dados[dados.duplicated(keep=False)]
```

Out[104…

	year_of_reference	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size	year_model	avg_price_brl
45791	2021.0	June	025232-8	5rtdwkpkpq5h	Renault	DUSTER OROCH Dyna. 2.0 Flex 16V Mec.	Gasoline	manual	2	2018.0	69893.0
45793	2021.0	June	025232-8	5rtdwkpkpq5h	Renault	DUSTER OROCH Dyna. 2.0 Flex 16V Mec.	Gasoline	manual	2	2018.0	69893.0
189895	2022.0	December	003296-4	3r6c277cnqcb	Ford	Ranger Limited 3.0 PSE 4x4 CD TB Diesel	Diesel	manual	3	2007.0	64638.0
189896	2022.0	December	003296-4	3r6c277cnqcb	Ford	Ranger Limited 3.0 PSE 4x4 CD TB Diesel	Diesel	manual	3	2007.0	64638.0

In [105…

```
# Removendo valores duplicados
dados.drop_duplicates(inplace=True)
```

In [106…

```
# Verifica se foram eliminados
dados.duplicated().sum()
```

Out[106… `np.int64(0)`

d. Crie duas categorias, para separar colunas numéricas e categóricas. Imprima o resumo de informações das variáveis numéricas e categóricas (estatística descritiva dos dados)

In [107…

```
# Verifica tipo dos dados
dados.dtypes
```

Out[107…
```
year_of_reference     float64
month_of_reference     object
fipe_code              object
authentication         object
brand                  object
model                  object
fuel                   object
gear                   object
engine_size            object
year_model            float64
avg_price_brl         float64
dtype: object
```

In [108…
```
# Transforma em inteiro
dados['year_of_reference'] = dados['year_of_reference'].astype(int)
dados['year_model'] = dados['year_model'].astype(int)
dados.dtypes
```

Out[108…
```
year_of_reference       int64
month_of_reference     object
fipe_code              object
authentication         object
brand                  object
model                  object
fuel                   object
gear                   object
engine_size            object
year_model              int64
avg_price_brl         float64
dtype: object
```

In [109…
```
numericas_cols = [col for col in dados.columns if dados[col].dtype != 'object']
categoricas_cols = [col for col in dados.columns if dados[col].dtype == 'object']
```

In [110…
```
dados[numericas_cols].describe()
```

Out[110…

	year_of_reference	year_model	avg_price_brl
count	202295.000000	202295.000000	202295.000000
mean	2021.564695	2011.271514	52756.765713
std	0.571904	6.376241	51628.912116
min	2021.000000	2000.000000	6647.000000
25%	2021.000000	2006.000000	22855.000000
50%	2022.000000	2012.000000	38027.000000
75%	2022.000000	2016.000000	64064.000000
max	2023.000000	2023.000000	979358.000000

In [111…

```
dados[categoricas_cols].describe()
```

Out[111…

	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size
count	202295	202295	202295	202295	202295	202295	202295	202295
unique	12	2091	202295	6	2112	3	2	29
top	January	003281-6	cfzlctzfwrcp	Fiat	Palio Week. Adv/Adv TRYON 1.8 mpi Flex	Gasoline	manual	1,6
freq	24260	425	1	44962	425	168684	161883	47420

e. Imprima a contagem de valores por modelo (model) e marca do carro (brand)

In [112…

```
 # Imprima a contagem de valores por modelo (model) e marca do carro (brand) - agrupados
contagem_marca_modelo = dados.groupby(['model', 'brand']).size().reset_index(name='count')

# Ordenar do mais frequente para o menos frequente
contagem_marca_modelo = contagem_marca_modelo.sort_values(by='count', ascending=False)
contagem_marca_modelo.head(10)
```

Out[112…

	model	brand	count
1248	Palio Week. Adv/Adv TRYON 1.8 mpi Flex	Fiat	425
547	Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p	Ford	425
1661	Saveiro 1.6 Mi/ 1.6 Mi Total Flex 8V	VW - VolksWagen	400
550	Focus 2.0 16V/SE/SE Plus Flex 5p Aut.	Ford	400
346	Doblo Adv/Adv TRYON/LOCKER 1.8 Flex	Fiat	375
750	Golf 2.0/ 2.0 Mi Flex Aut/Tiptronic.	VW - VolksWagen	375
301	Corvette 5.7/ 6.0, 6.2 Targa/Stingray	GM - Chevrolet	375
307	Courier XL/XL-RS 1.6/ XL 1.6 Flex	Ford	350
918	Kombi Escolar 1.6 MPi	VW - VolksWagen	350
303	Courier 1.6 L/ 1.6 Flex	Ford	350

---

2. Visualização dos dados

a. Gere um gráfico da distribuição da quantidade de carros por marca

In [113…

```
quantidade_marca = dados['brand'].value_counts()
print(quantidade_marca)
```

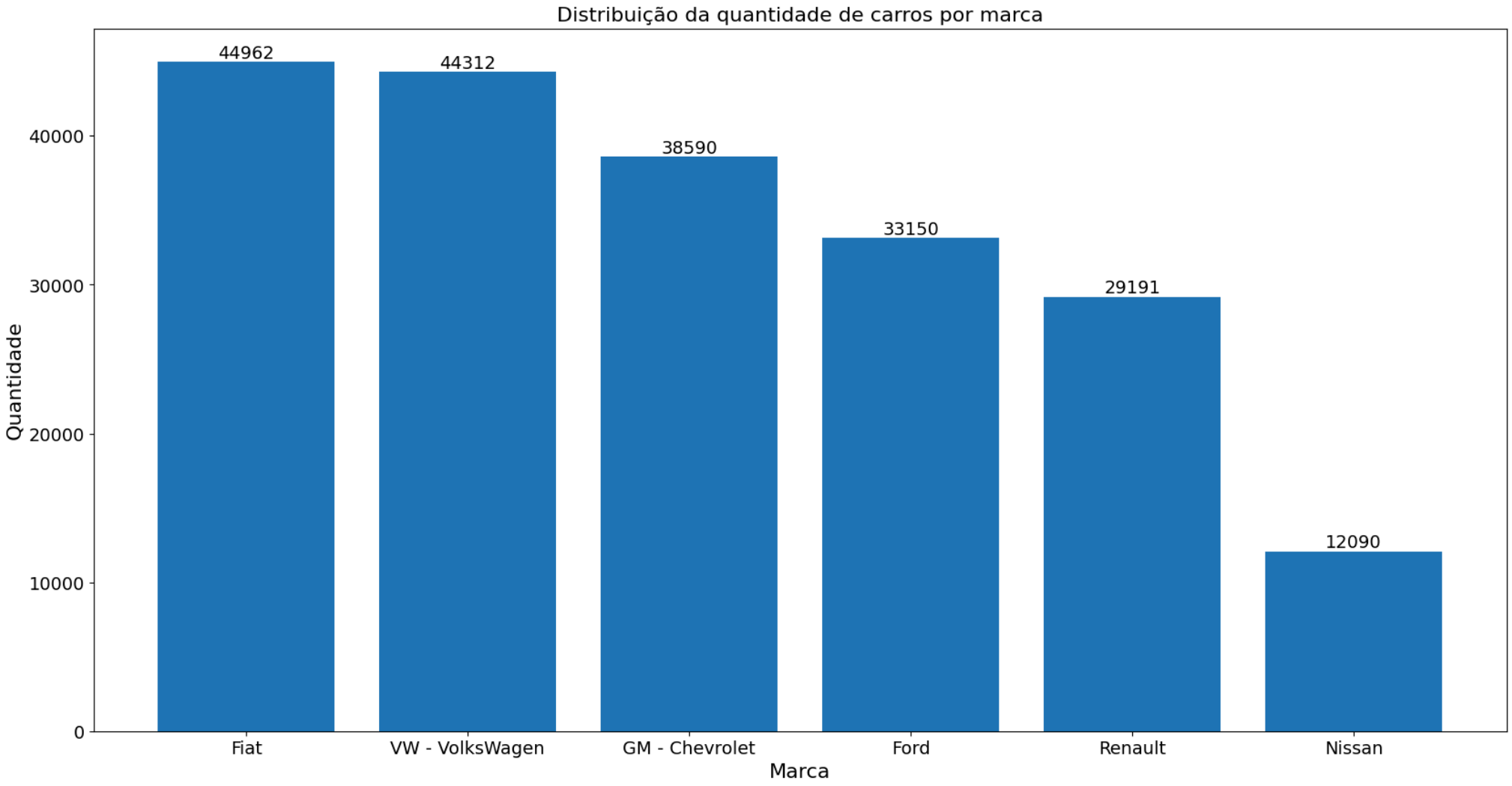
```
brand
Fiat               44962
VW - VolksWagen    44312
GM - Chevrolet     38590
Ford               33150
Renault            29191
Nissan             12090
Name: count, dtype: int64
```

In [114…

```
# Quantidade de carros por marca
plt.figure(figsize=(20,10))
grafico_1 = plt.bar(quantidade_marca.index, quantidade_marca.values) # Variavel quantidade_marca eixo X
plt.title('Distribuição da quantidade de carros por marca', fontsize=16) # Inserção do título
plt.xlabel('Marca', fontsize=16) # Rótulo do eixo Y
```

```
plt.ylabel('Quantidade', fontsize=16) # Rótulo do eixo Y
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)

plt.bar_label(grafico_1, size=14);
```



b. Gere um gráfico da distribuição da quantidade de carros por tipo de engrenagem do carro

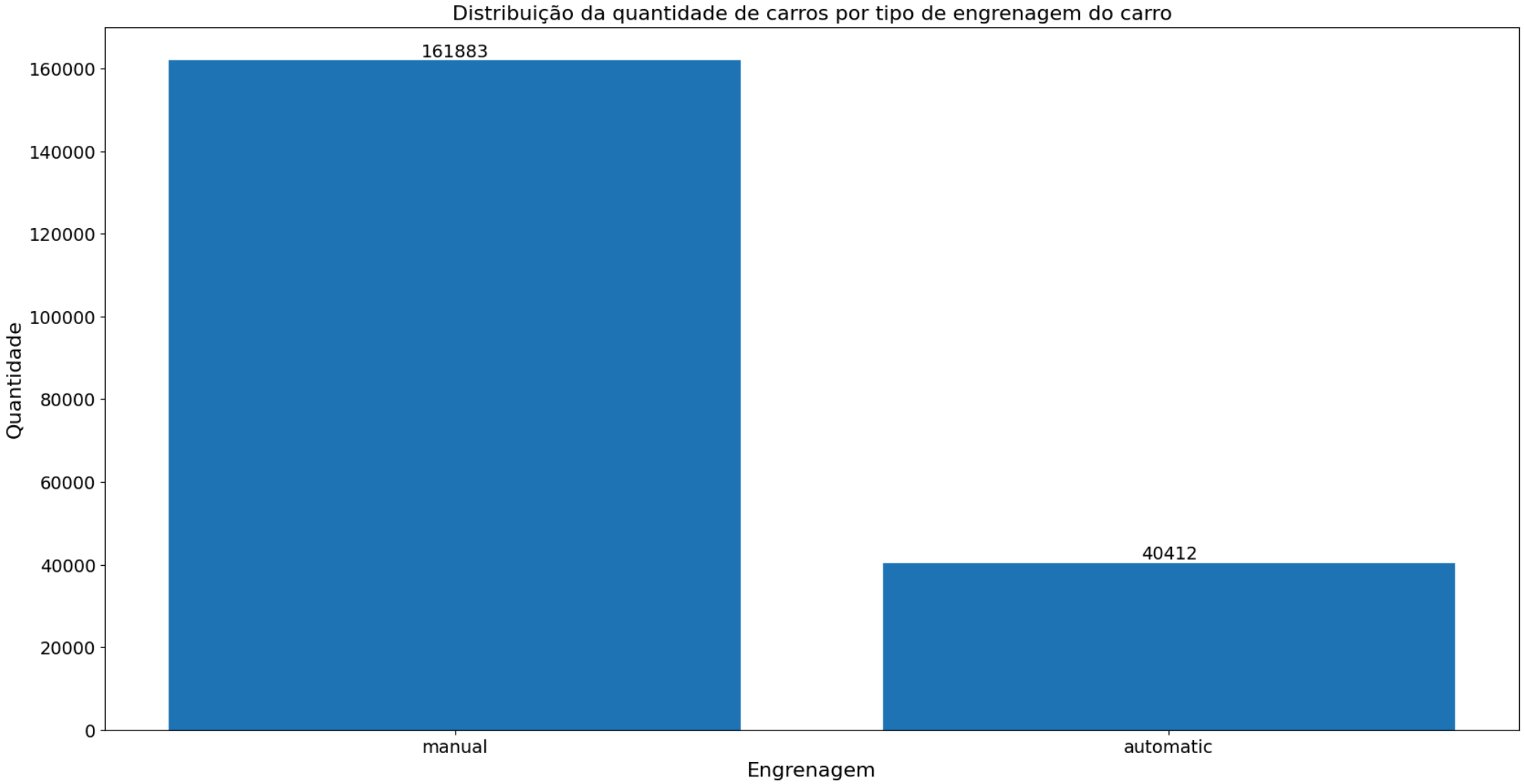
In [115…

```
# totalizar
quantidade_engrenagem = dados['gear'].value_counts()
print(quantidade_engrenagem)
```

```
gear
manual       161883
automatic     40412
Name: count, dtype: int64
```

In [116…

```python
# Gerar Gráfico
plt.figure(figsize=(20,10))
grafico_2 = plt.bar(quantidade_engrenagem.index, quantidade_engrenagem.values) # Variavel quantidade_marca eixo X
plt.title('Distribuição da quantidade de carros por tipo de engrenagem do carro', fontsize=16) # Inserção do título
plt.xlabel('Engrenagem', fontsize=16) # Rótulo do eixo Y
plt.ylabel('Quantidade', fontsize=16) # Rótulo do eixo Y
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.bar_label(grafico_2, size=14);
```



c. Gere um gráfico da evolução da média de preço dos carros ao longo dos meses de 2022 (variável de tempo no eixo X)

In [117…
```
# Verifica tipo dos dados
dados.dtypes
```

Out[117…
```
year_of_reference       int64
month_of_reference     object
fipe_code              object
authentication         object
brand                  object
model                  object
fuel                   object
gear                   object
engine_size            object
year_model              int64
avg_price_brl         float64
dtype: object
```

In [118…
```
# Cria coluna com o mês (numero)
dados['Mês de referência'] = pd.to_datetime(dados['month_of_reference'], format='%B').dt.month
```

In [119…
```
dados.dtypes
```

Out[119…
```
year_of_reference       int64
month_of_reference     object
fipe_code              object
authentication         object
brand                  object
model                  object
fuel                   object
gear                   object
engine_size            object
year_model              int64
avg_price_brl         float64
Mês de referência       int32
dtype: object
```

In [120…
```
# Filtrar apenas os dados do ano de 2022
dados_2022 = dados[dados['year_of_reference'] == 2022]
dados_2022.head()
```

Out[120…

	year_of_reference	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size	year_model	avg_price_brl	ref
96280	2022	January	004001-0	gzw0hkct8cj4	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2002	12330.0	
96281	2022	January	004001-0	gm2ws5yqjnfx	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2001	11408.0	
96282	2022	January	004001-0	gbvgy7432kp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2000	10620.0	
96283	2022	January	004001-0	gvx412fg8v0	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Alcohol	manual	1	2000	11992.0	
96284	2022	January	004003-7	jtskpmg524fx	GM - Chevrolet	Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI	Gasoline	manual	1,6	2001	17182.0	

In [121…

```
media_preco_ano = dados_2022.groupby(['Mês de referência'])['avg_price_brl'].mean().round(0)
media_preco_ano.head()
```

Out[121…

```
Mês de referência
1    54840.0
2    55825.0
3    56849.0
4    57150.0
5    57800.0
Name: avg_price_brl, dtype: float64
```

In [122…

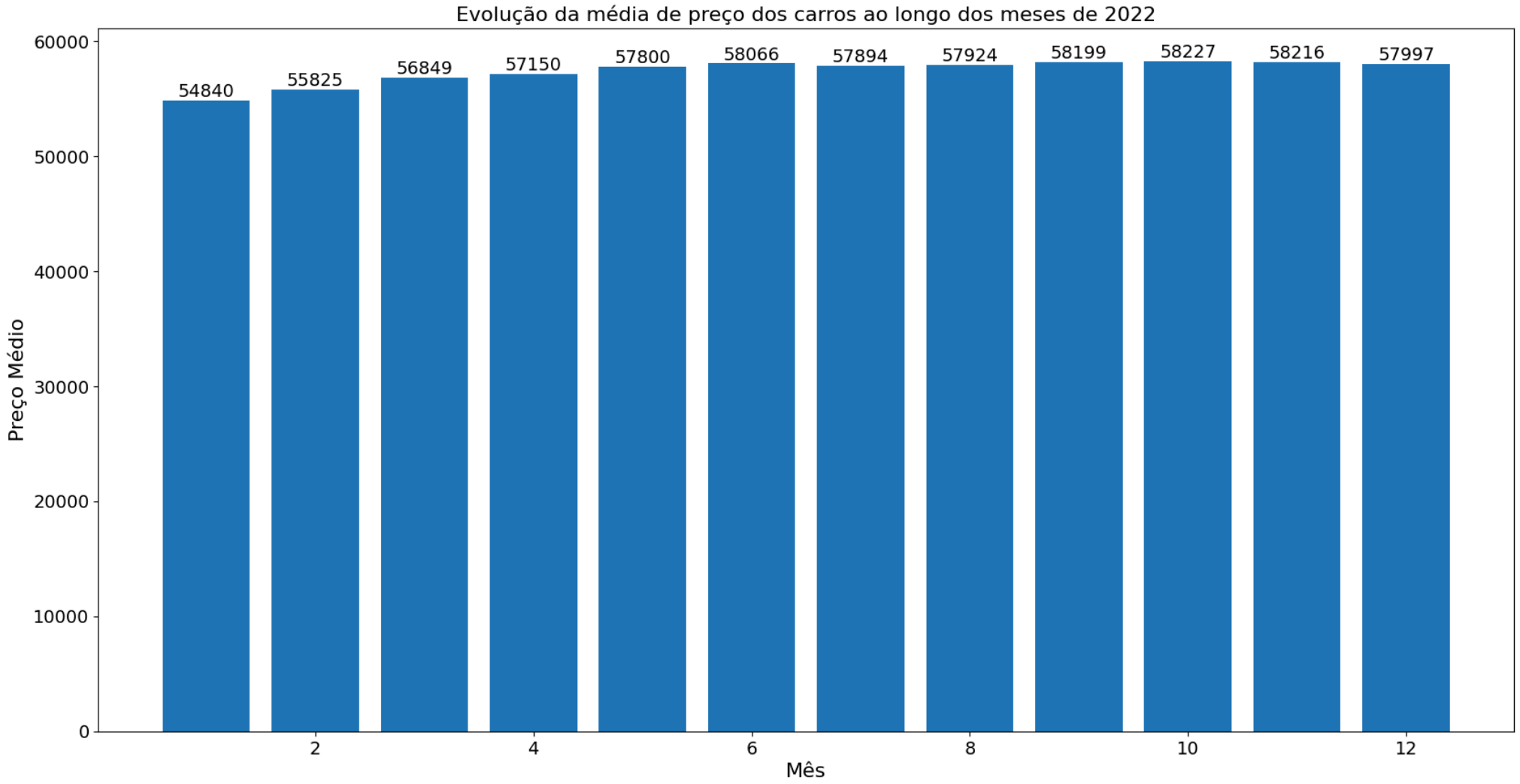
```
media_preco_ano = media_preco_ano.reset_index(name='Media de precos')
media_preco_ano.head()
```

Out[122…

	Mês de referência	Media de precos
0	1	54840.0
1	2	55825.0
2	3	56849.0
3	4	57150.0
4	5	57800.0

In [123…

```
# Gerar Grafico
plt.figure(figsize=(20,10))
grafico_3 = plt.bar(media_preco_ano['Mês de referência'], media_preco_ano['Media de precos']) # Variavel quantidade_marca eixo X
plt.title('Evolução da média de preço dos carros ao longo dos meses de 2022', fontsize=16) # Inserção do título
plt.xlabel('Mês', fontsize=16) # Rótulo do eixo Y
plt.ylabel('Preço Médio', fontsize=16) # Rótulo do eixo Y
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.bar_label(grafico_3, size=14);
```



d. Gere um gráfico da distribuição da média de preço dos carros por marca e tipo de engrenagem

In [124…
```
media_preco_marca_engrenagem = dados.groupby(['brand', 'gear'])['avg_price_brl'].mean().round(0)
media_preco_marca_engrenagem.head()
```

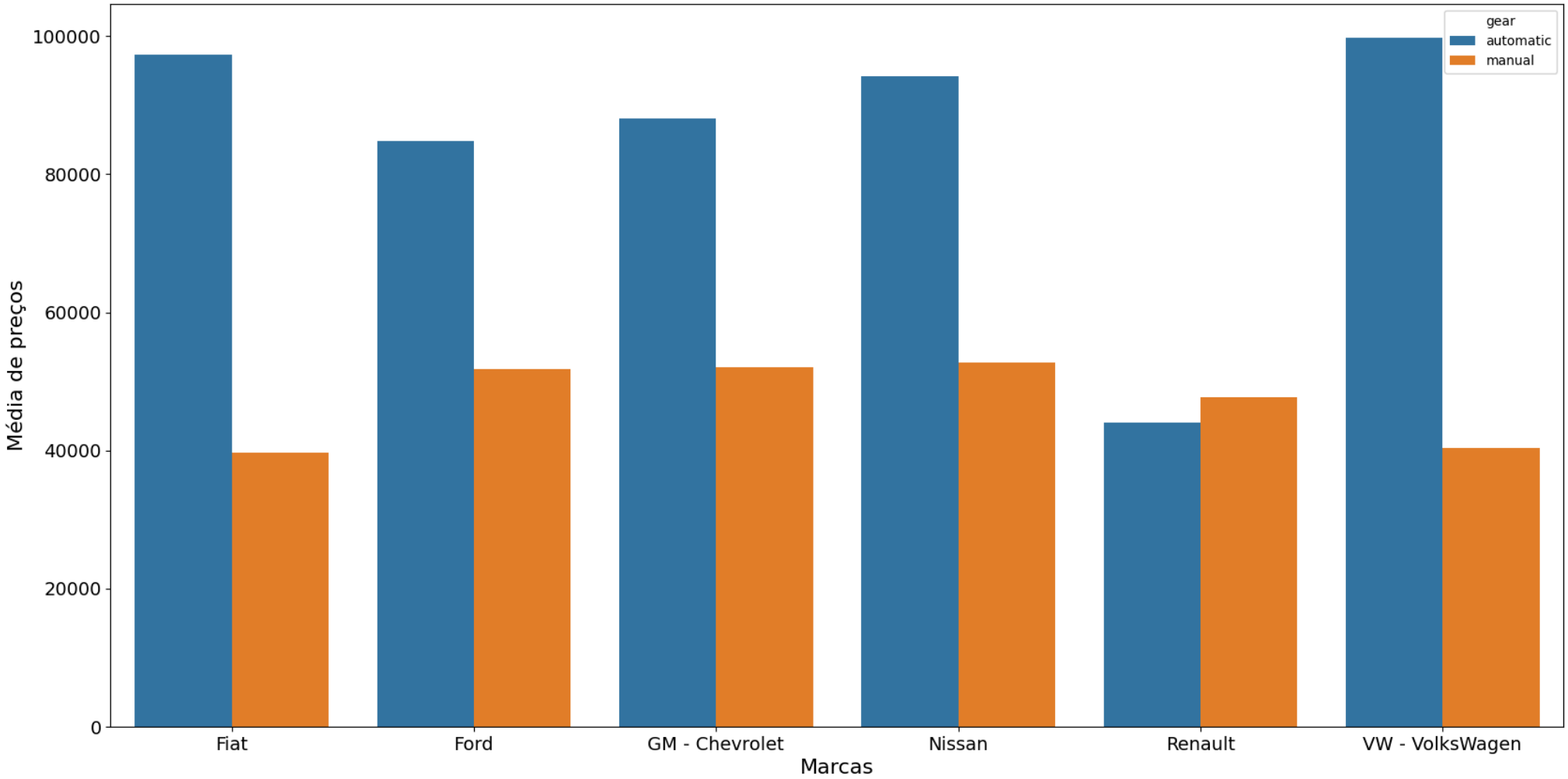
Out[124…
```
brand          gear
Fiat           automatic    97397.0
               manual       39694.0
Ford           automatic    84769.0
               manual       51784.0
GM - Chevrolet automatic    88157.0
Name: avg_price_brl, dtype: float64
```

In [125…
```
media_preco_marca_engrenagem = media_preco_marca_engrenagem.reset_index(name='Media de precos')
media_preco_marca_engrenagem.head()
```

Out[125…

	brand	gear	Media de precos
0	Fiat	automatic	97397.0
1	Fiat	manual	39694.0
2	Ford	automatic	84769.0
3	Ford	manual	51784.0
4	GM - Chevrolet	automatic	88157.0

In [126…
```
plt.figure(figsize=(20,10))
sns.barplot(x='brand', y='Media de precos', hue='gear', data=media_preco_marca_engrenagem, hue_order=['automatic', 'manual'])
plt.ylabel('Média de preços', fontsize=16)
plt.xlabel('Marcas', fontsize=16)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14);
```



f. Gere um gráfico da distribuição da média de preço dos carros por marca e tipo de combustível

In [127…
```
media_preco_marca_combustivel = dados.groupby(['brand', 'fuel'])['avg_price_brl'].mean().round(0)
media_preco_marca_combustivel.head()
```

Out[127…
```
brand  fuel
Fiat   Alcohol     11510.0
       Diesel      99814.0
       Gasoline    37197.0
Ford   Alcohol     10149.0
       Diesel      94526.0
Name: avg_price_brl, dtype: float64
```

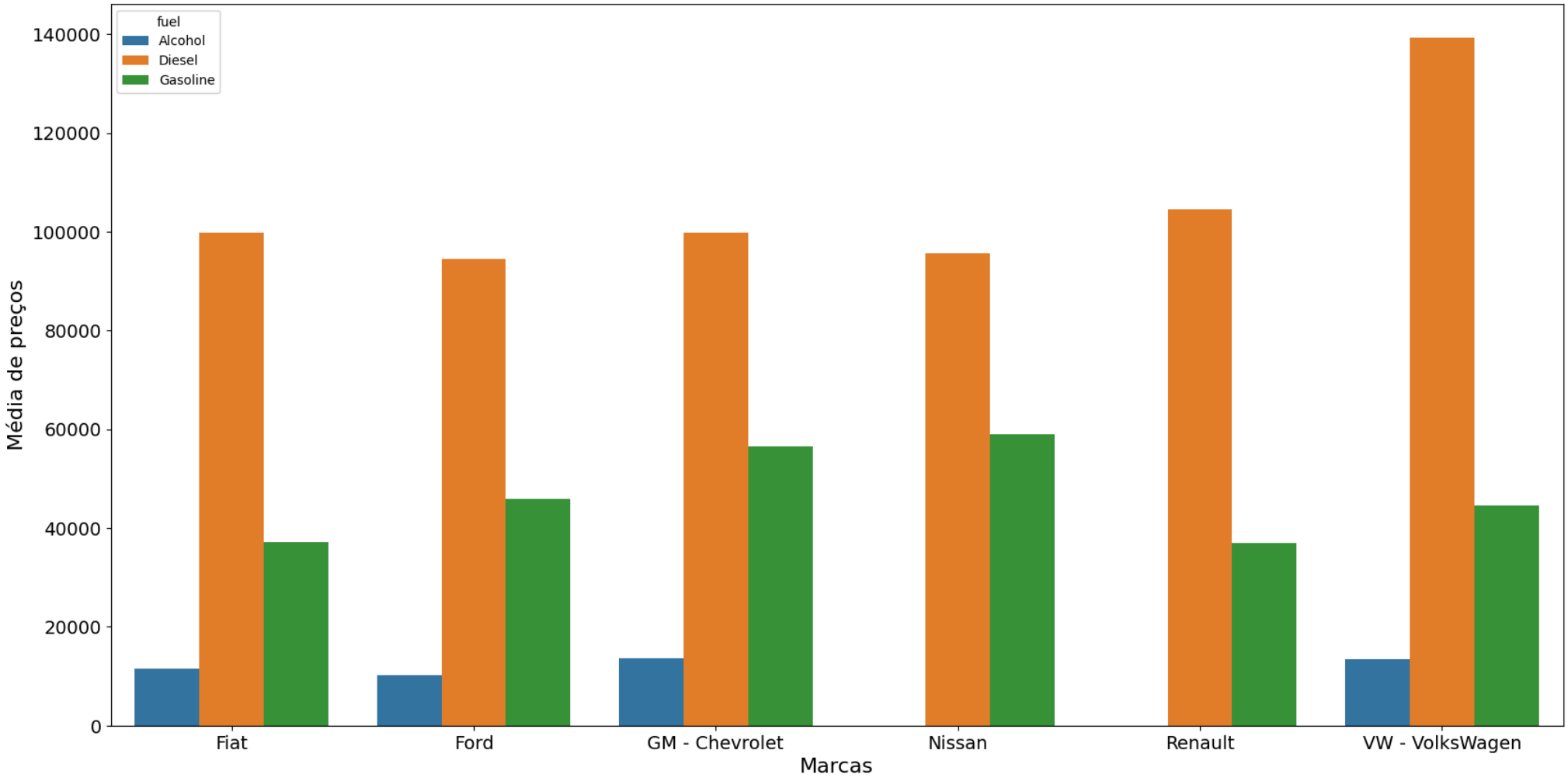
In [128…
```
media_preco_marca_combustivel = media_preco_marca_combustivel.reset_index(name='Media de precos')
media_preco_marca_combustivel.head()
```

Out[128…

	brand	fuel	Media de precos
0	Fiat	Alcohol	11510.0
1	Fiat	Diesel	99814.0
2	Fiat	Gasoline	37197.0
3	Ford	Alcohol	10149.0
4	Ford	Diesel	94526.0

In [129…
```
plt.figure(figsize=(20,10))
sns.barplot(x='brand', y='Media de precos', hue='fuel', data=media_preco_marca_combustivel, hue_order=['Alcohol', 'Diesel', 'Gasoline'])

plt.ylabel('Média de preços', fontsize=16)
plt.xlabel('Marcas', fontsize=16)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14);
```



---

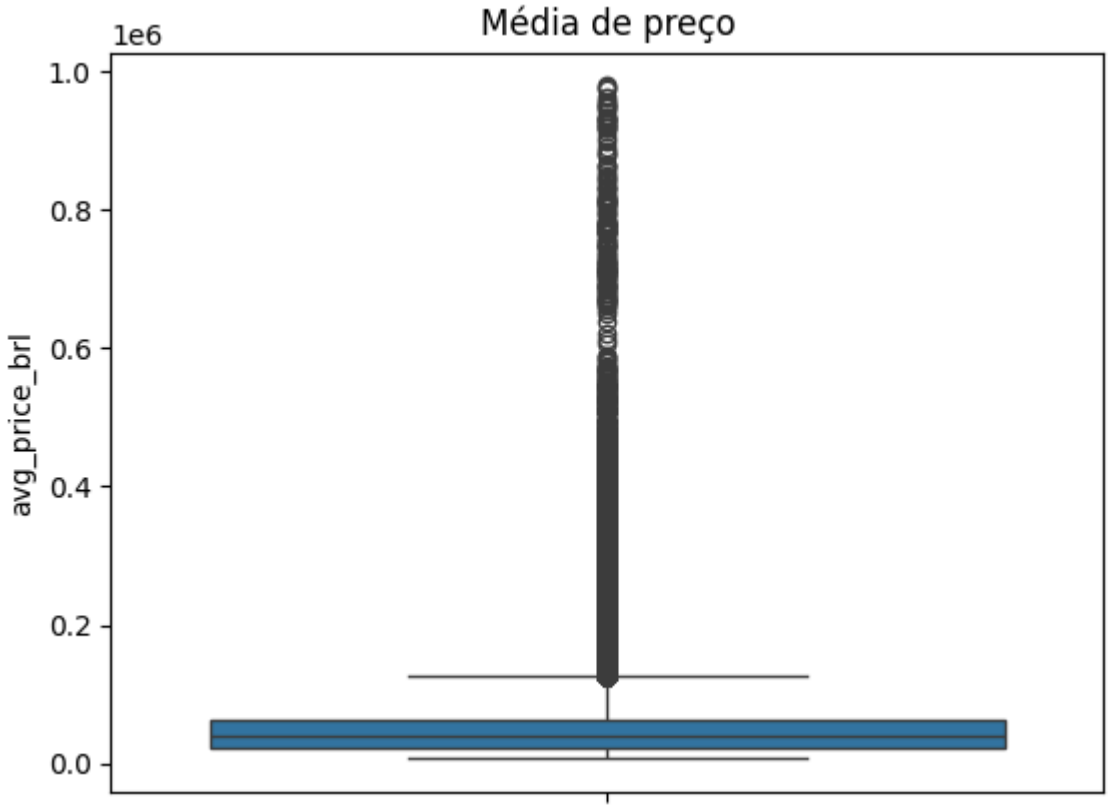
3. Aplicação de modelos de machine learning para prever o preço médio dos carros

a. Escolha as variáveis numéricas (modelos de Regressão) para serem as variáveis independentes do modelo.A variável target é avg_price.

In [130…

```
# Análise da variável target
sns.boxplot(dados['avg_price_brl']).set_title("Média de preço")
```

Out[130… Text(0.5, 1.0, 'Média de preço')



In [131…

```
dados.head()
```

Out[131…

	year_of_reference	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size	year_model	avg_price_brl	Mês referên
0	2021	January	004001-0	cfzlctzfwrcp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2002	9162.0	
1	2021	January	004001-0	cdqwxwpw3y2p	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2001	8832.0	
2	2021	January	004001-0	cb1t3xwwj1xp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2000	8388.0	
3	2021	January	004001-0	cb9gct6j65r0	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Alcohol	manual	1	2000	8453.0	
4	2021	January	004003-7	g15wg0gbz1fx	GM - Chevrolet	Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI	Gasoline	manual	1,6	2001	12525.0	

In [132…

```
# Transformação da variável gear para numérica
dados['gear'] = LabelEncoder().fit_transform(dados['gear'])
dados.head()
```

Out[132…

	year_of_reference	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size	year_model	avg_price_brl	Mês de referência
0	2021	January	004001-0	cfzlctzfwrcp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	1	1	2002	9162.0	1
1	2021	January	004001-0	cdqwxwpw3y2p	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	1	1	2001	8832.0	1
2	2021	January	004001-0	cb1t3xwwj1xp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	1	1	2000	8388.0	1
3	2021	January	004001-0	cb9gct6j65r0	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Alcohol	1	1	2000	8453.0	1
4	2021	January	004003-7	g15wg0gbz1fx	GM - Chevrolet	Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI	Gasoline	1	1,6	2001	12525.0	1

In [133…

```
# Transformação da variável fuel para numérica
dados['fuel'] = LabelEncoder().fit_transform(dados['fuel'])
dados.head()
```

Out[133…

	year_of_reference	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size	year_model	avg_price_brl	Mês de referência
0	2021	January	004001-0	cfzlctzfwrcp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2002	9162.0	1
1	2021	January	004001-0	cdqwxwpw3y2p	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2001	8832.0	1
2	2021	January	004001-0	cb1t3xwwj1xp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2000	8388.0	1
3	2021	January	004001-0	cb9gct6j65r0	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	0	1	1	2000	8453.0	1
4	2021	January	004003-7	g15wg0gbz1fx	GM - Chevrolet	Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI	2	1	1,6	2001	12525.0	1

In [134…

```
# Transformando coluna brand em número
# Função para atribuir valores numéricos com base na marca
def atribuir_valor_numerico(id_brand):

    if id_brand == 'Fiat':
        return 10
    elif id_brand == 'Ford':
        return 20
    elif id_brand == 'GM - Chevrolet':
        return 30
    elif id_brand == 'Nissan':
```

```
        return 40
    elif id_brand == 'Renault':
        return 50
    elif id_brand == 'VW - WolksWagen':
        return 60
    else:
        return None

# Criar a nova coluna usando a função aplicada na coluna 'Anos experiencia'
dados['brand - numerico'] = dados['brand'].apply(atribuir_valor_numerico)

# Nome das colunas
dados.columns
```

Out[134…
```
Index(['year_of_reference', 'month_of_reference', 'fipe_code',
       'authentication', 'brand', 'model', 'fuel', 'gear', 'engine_size',
       'year_model', 'avg_price_brl', 'Mês de referência', 'brand - numerico'],
      dtype='object')
```

In [135…
```
dados['engine_size'].unique()
```

Out[135…
```
array(['1', '1,6', '2,2', '4,3', '2,5', '1,8', '2', '4,2', '3,8', '4,1',
       '5,7', '2,8', '2,4', '1,4', '3,6', '6,2', '3', '1,2', '1,5', '1,3',
       '1,9', '2,3', '4', '3,9', '5', '3,5', '3,2', '2,7', '3,3'],
      dtype=object)
```

In [136…
```
dados['engine_size'] = dados['engine_size'].astype(str).str.replace(',', '.').astype(float)
dados['engine_size'].unique()
```

Out[136…
```
array([1. , 1.6, 2.2, 4.3, 2.5, 1.8, 2. , 4.2, 3.8, 4.1, 5.7, 2.8, 2.4,
       1.4, 3.6, 6.2, 3. , 1.2, 1.5, 1.3, 1.9, 2.3, 4. , 3.9, 5. , 3.5,
       3.2, 2.7, 3.3])
```

In [137…
```
dados.dtypes
```

Out[137…
```
year_of_reference       int64
month_of_reference     object
fipe_code              object
authentication         object
brand                  object
model                  object
fuel                    int64
gear                    int64
engine_size           float64
year_model              int64
avg_price_brl         float64
Mês de referência       int32
brand - numerico      float64
dtype: object
```

In [138…
```
# Variável dados_num apenas com colunas que são de interesse
dados_num = dados.drop(['month_of_reference','Mês de referência','fipe_code', 'authentication', 'brand', 'model'], axis = 1)
dados_num.head()
```

Out[138…

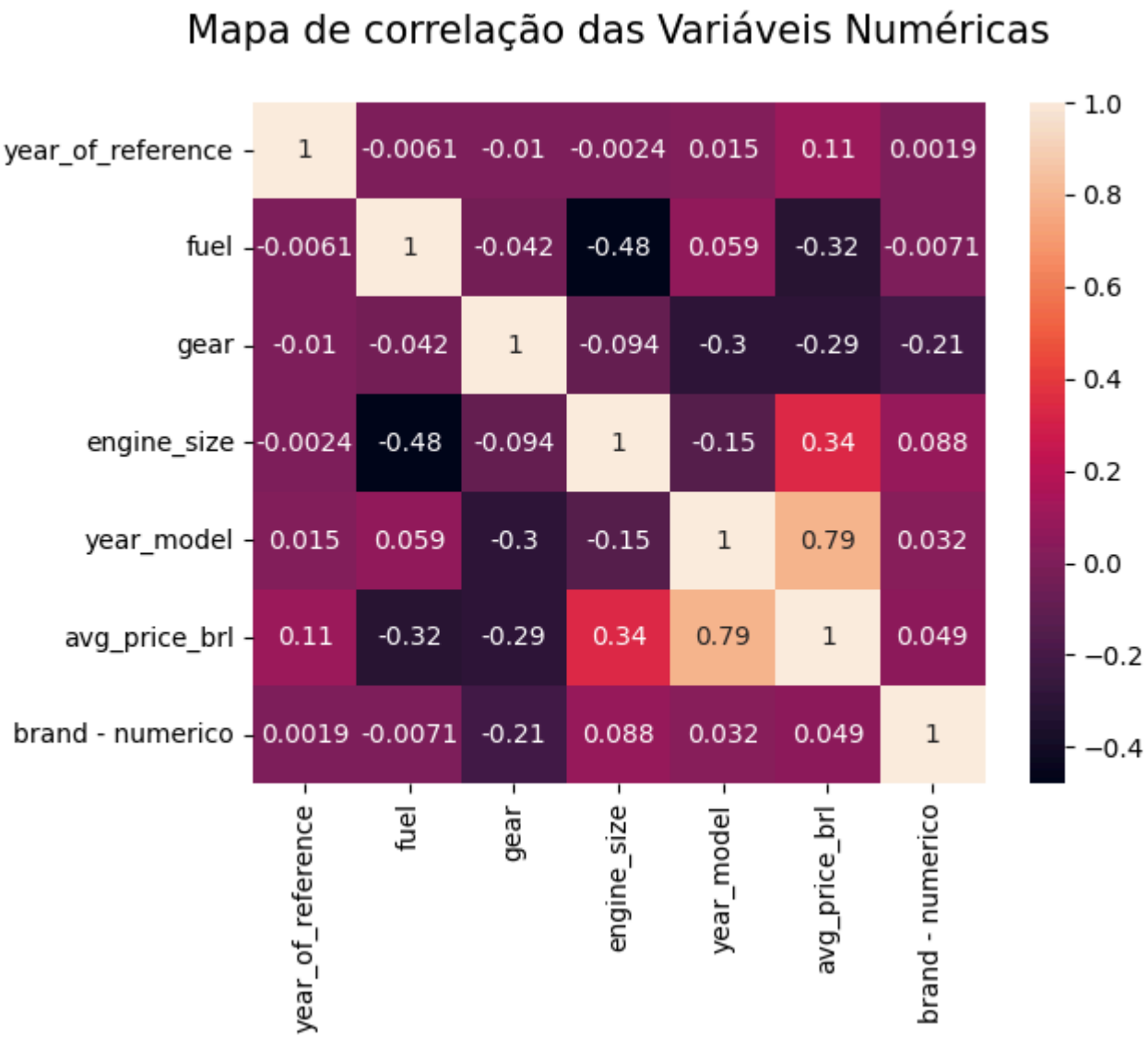
	year_of_reference	fuel	gear	engine_size	year_model	avg_price_brl	brand - numerico
0	2021	2	1	1.0	2002	9162.0	30.0
1	2021	2	1	1.0	2001	8832.0	30.0
2	2021	2	1	1.0	2000	8388.0	30.0
3	2021	0	1	1.0	2000	8453.0	30.0
4	2021	2	1	1.6	2001	12525.0	30.0

In [139…
```
dados_num.dtypes
```

Out[139…
```
year_of_reference      int64
fuel                   int64
gear                   int64
engine_size          float64
year_model             int64
avg_price_brl        float64
brand - numerico     float64
dtype: object
```

b. Crie partições contendo 75% dos dados para treino e 25% para teste

In [140…]
```
# Mapa de correlação das variáveis numéricas com variável target
sns.heatmap(dados_num.corr("spearman"), annot=True)
plt.title("Mapa de correlação das Variáveis Numéricas\n", fontsize = 15)
plt.show()
```



In [141…]
```
# Variável X contém apenas variáveis numéricas de interesse para a análise, excluindo a variável target
X = dados_num.drop(['avg_price_brl'], axis = 1)
X.head()
```

Out[141…

	year_of_reference	fuel	gear	engine_size	year_model	brand - numerico
0	2021	2	1	1.0	2002	30.0
1	2021	2	1	1.0	2001	30.0
2	2021	2	1	1.0	2000	30.0
3	2021	0	1	1.0	2000	30.0
4	2021	2	1	1.6	2001	30.0

In [142…
```
X.dtypes
```

Out[142…
```
year_of_reference      int64
fuel                   int64
gear                   int64
engine_size          float64
year_model             int64
brand - numerico     float64
dtype: object
```

In [143…
```
# Variável Y contém apenas a variável target - Faixa Salarial
Y = dados_num['avg_price_brl']
Y.head()
```

Out[143…
```
0     9162.0
1     8832.0
2     8388.0
3     8453.0
4    12525.0
Name: avg_price_brl, dtype: float64
```

In [144…
```
# Divisão: 25% dos dados são de teste e 75% de treinamento
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=42)
```

In [145…
```
# Observando os dados de treinamento
print(X_train.shape)
X_train.head(1)
```

```
(151721, 6)
```

Out[145…

	year_of_reference	fuel	gear	engine_size	year_model	brand - numerico
156364	2022	1	1	2.3	2020	10.0

In [146…
```
X_train.dtypes
```

Out[146…
```
year_of_reference      int64
fuel                   int64
gear                   int64
engine_size          float64
year_model             int64
brand - numerico     float64
dtype: object
```

In [147…
```
# Observando os dados de teste
print( X_test.shape)
X_test.head(1)
```

```
(50574, 6)
```

Out[147…

	year_of_reference	fuel	gear	engine_size	year_model	brand - numerico
180633	2022	2	1	1.6	2015	10.0

In [148…
```
# Observando a variável target
Y_test.head()
```

Out[148…
```
180633    42595.0
13130     10989.0
163315     9087.0
121464    26965.0
14044     57102.0
Name: avg_price_brl, dtype: float64
```

c. Treine modelos RandomForest (biblioteca RandomForestRegressor) e XGBoost (biblioteca XGBRegressor) para predição dos preços dos carros.

RandomForest sem parâmetros

In [149…
```
# Algoritmo Random Forest, sem especificar nenhum parâmetro (número de árvores, número de ramificações, etc)
model_rf = RandomForestRegressor()
```

In [150…
```
X_train.dtypes
```

Out[150…
```
year_of_reference      int64
fuel                   int64
gear                   int64
engine_size          float64
year_model             int64
brand - numerico     float64
dtype: object
```

In [151…
```
# Ajuste do modelo, de acordo com as variáveis de treinamento
model_rf.fit(X_train, Y_train)
```

Out[151…
```
▾ RandomForestRegressor
RandomForestRegressor()
```

d. Grave os valores preditos em variáveis criadas

In [152…
```
# Predição dos valores de média de preço com base nos dados de teste
valores_preditos_rf = model_rf.predict(X_test)
```

In [153…
```
# Valores preditos
valores_preditos_rf
```

Out[153…
```
array([ 44901.40940087,  12761.19018077,  15296.50429386, ...,
       116684.92106355,  16213.26076138,  21535.89805778], shape=(50574,))
```

e. Realize a análise de importância das variáveis para estimar a variável target, para cada modelo treinado

In [154…
```
model_rf.feature_importances_
feature_importances = pd.DataFrame(model_rf.feature_importances_, index=X_train.columns, columns=['importance']).sort_values('importance',
feature_importances
```

Out[154...

	importance
engine_size	0.479723
year_model	0.407759
gear	0.036041
fuel	0.033603
brand - numerico	0.029169
year_of_reference	0.013706

g. Escolha o melhor modelo com base nas métricas de avaliação MSE, MAE e R^2

In [155...

```
mse = mean_squared_error(Y_test, valores_preditos_rf)
mae = mean_absolute_error(Y_test, valores_preditos_rf)
r2_score(Y_test, valores_preditos_rf)
```

Out[155...

```
0.9577581044218859
```

Modelo Random Forest com parâmetros

In [156...

```
model_rf_parametros = RandomForestRegressor(max_depth=29, min_samples_leaf=32, min_samples_split=28, n_estimators=208, random_state=43)
```

In [157...

```
X_train.dtypes
```

Out[157...

```
year_of_reference      int64
fuel                   int64
gear                   int64
engine_size          float64
year_model             int64
brand - numerico     float64
dtype: object
```

In [158...

```
# Ajuste do modelo, de acordo com as variáveis de treinamento
model_rf_parametros.fit(X_train, Y_train)
```

Out[158…

RandomForestRegressor

```
RandomForestRegressor(max_depth=29, min_samples_leaf=32, min_samples_split=28,
                      n_estimators=208, random_state=43)
```

d. Grave os valores preditos em variáveis criadas

In [159…

```
# Predição dos valores de média de preço com base nos dados de teste
valores_preditos_rf_parametros = model_rf_parametros.predict(X_test)
pd.DataFrame(valores_preditos_rf).round(1)
```

Out[159…

	0
0	44901.4
1	12761.2
2	15296.5
3	26053.6
4	70991.0
...	...
50569	14141.0
50570	35401.1
50571	116684.9
50572	16213.3
50573	21535.9

50574 rows × 1 columns

e. Realize a análise de importância das variáveis para estimar a variável target, para cada modelo treinado

In [160…

```
model_rf_parametros.feature_importances_
feature_importances = pd.DataFrame(model_rf_parametros.feature_importances_, index=X_train.columns, columns=['importance']).sort_values('im
feature_importances
```

Out[160…

	importance
engine_size	0.483297
year_model	0.422444
fuel	0.035435
gear	0.024156
brand - numerico	0.023270
year_of_reference	0.011400

g. Escolha o melhor modelo com base nas métricas de avaliação MSE, MAE e R^2

In [161…

```
mse = mean_squared_error(Y_test, valores_preditos_rf_parametros)
mae = mean_absolute_error(Y_test, valores_preditos_rf_parametros)
r2_score(Y_test, valores_preditos_rf_parametros)
```

Out[161…

```
0.9221806568158424
```

XGBoost

In [162…

```
model_xgboost = XGBRegressor()
```

In [163…

```
X_train.dtypes
```

Out[163…

```
year_of_reference      int64
fuel                   int64
gear                   int64
engine_size          float64
year_model             int64
brand - numerico     float64
dtype: object
```

In [164…

```
# Ajuste do modelo, de acordo com as variáveis de treinamento
model_xgboost.fit(X_train, Y_train)
```

Out[164…

XGBRegressor

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             feature_weights=None, gamma=None, grow_policy=None,
             importance_type=None, interaction_constraints=None,
             learning_rate=None, max_bin=None, max_cat_threshold=None,
             max_cat_to_onehot=None, max_delta_step=None, max_depth=None,
             max_leaves=None, min_child_weight=None, missing=nan,
```

d. Grave os valores preditos em variáveis criadas

In [165…

```
# Predição dos valores de salário com base nos dados de teste
valores_preditos_xgboost = model_xgboost.predict(X_test)
valores_preditos_xgboost
```

Out[165…

```
array([ 45431.6  ,  12832.293,  15686.486, ..., 117207.42 ,  15483.117,
        21800.822], shape=(50574,), dtype=float32)
```

e. Realize a análise de importância das variáveis para estimar a variável target, para cada modelo treinado

In [166…

```
model_xgboost.feature_importances_
feature_importances = pd.DataFrame(model_xgboost.feature_importances_, index=X_train.columns, columns=['importance']).sort_values('importan
feature_importances
```

Out[166…

	importance
engine_size	0.401869
year_model	0.245797
gear	0.165223
fuel	0.137706
brand - numerico	0.031376
year_of_reference	0.018029

g. Escolha o melhor modelo com base nas métricas de avaliação MSE, MAE e R^2

In [167…

```
mse = mean_absolute_error(Y_test, valores_preditos_xgboost)
mae = mean_absolute_error(Y_test, valores_preditos_xgboost)
r2_score(Y_test, valores_preditos_xgboost)
```

Out[167…

```
0.9595925492647421
```