

Trabalho Final

Linguagem de Programação Aplicada - LPA

Cicero Samuel Mendes

1 Carregando Bibliotecas

```
[117]: import numpy as np
import pandas as pd
import seaborn as sns

import matplotlib as mpl
import matplotlib.pyplot as plt
plt.style.use('ggplot');
mpl.rcParams['figure.dpi'] = 300

import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from xgboost import XGBRegressor
```

2 Carregando Dados

```
[2]: dados = pd.read_csv('datasets/precos_carros_brasil.csv')
```

```
[3]: dados.columns
```

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

```
[4]: dados.head()
```

```
[4]:   year_of_reference month_of_reference fipecode authentication \
0          2021.0          January  004001-0  cfzltzfwrcp
1          2021.0          January  004001-0  cdqwxwpw3y2p
2          2021.0          January  004001-0  cb1t3xwwj1xp
3          2021.0          January  004001-0  cb9gct6j65r0
4          2021.0          January  004003-7  g15wg0gbz1fx

          brand                                model    fuel
↳gear \
0  GM - Chevrolet          Corsa Wind 1.0 MPFI / EFI 2p  Gasoline
↳manual
1  GM - Chevrolet          Corsa Wind 1.0 MPFI / EFI 2p  Gasoline
↳manual
2  GM - Chevrolet          Corsa Wind 1.0 MPFI / EFI 2p  Gasoline
↳manual
3  GM - Chevrolet          Corsa Wind 1.0 MPFI / EFI 2p  Alcohol
↳manual
4  GM - Chevrolet  Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI  Gasoline
↳manual

          engine_size  year_model  avg_price_brl
0              1         2002.0         9162.0
1              1         2001.0         8832.0
2              1         2000.0         8388.0
3              1         2000.0         8453.0
4             1,6         2001.0        12525.0
```

```
[5]: dados.shape
```

```
[5]: (267542, 11)
```

3 Análise Exploratória dos Dados

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

```
[6]: # verificando se há valores ausentes
dados.isna().any()
```

```
[6]: year_of_reference      True
      month_of_reference    True
      fipecode              True
      authentication        True
      brand                 True
      model                 True
      fuel                  True
      gear                  True
      engine_size           True
      year_model            True
      avg_price_brl         True
      dtype: bool
```

```
[7]: # verificando a quantidade de valores ausentes
dados.isna().sum()
```

```
[7]: year_of_reference      65245
      month_of_reference    65245
      fipecode              65245
      authentication        65245
      brand                 65245
      model                 65245
      fuel                  65245
      gear                  65245
      engine_size           65245
      year_model            65245
      avg_price_brl         65245
      dtype: int64
```

```
[8]: # excluindo linhas com valores ausentes
dados.dropna(axis='index', inplace=True)
```

```
[9]: dados.shape
```

```
[9]: (202297, 11)
```

- Verifique se há dados duplicados nos dados

```
[10]: # verificando se há dados duplicados
dados.duplicated()
```

```
[10]: 0      False
      1      False
      2      False
      3      False
      4      False
      ...
      202292  False
      202293  False
      202294  False
      202295  False
      202296  False
      Length: 202297, dtype: bool
```

O dados não possuem valores duplicados.

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

```
[11]: # criando categorias para separar colunas numéricas e categóricas
cols_numericas = [col for col in dados.columns if dados[col].dtype !=
    ↳ 'object']
cols_categoricas = [col for col in dados.columns if dados[col].dtype ==
    ↳ 'object']
```

```
[12]: # resumo das variáveis numéricas
dados[cols_numericas].describe()
```

```
[12]:
```

	year_of_reference	year_model	avg_price_brl
count	202297.000000	202297.000000	202297.000000
mean	2021.564694	2011.271527	52756.909153
std	0.571903	6.376234	51628.677716
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

```
[13]: # resumo das variáveis categóricas
dados[cols_categoricas].describe()
```

```
[13]:
```

	month_of_reference	fipe_code	authentication	brand \
count	202297	202297	202297	202297
unique	12	2091	202295	6
top	January	003281-6	3r6c277cnqcb	Fiat
freq	24260	425	2	44962

	model	fuel	gear	
→engine_size				
count	202297	202297	202297	↳
→202297				
unique	2112	3	2	↳
→29				
top	Palio Week. Adv/Adv TRYON 1.8 mpi Flex	Gasoline	manual	↳
→1,6				
freq	425	168685	161885	↳
→47420				

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

```
[15]: # contagem dos modelos por modelo
dados['model'].value_counts()
```

```
[15]: model
Palio Week. Adv/Adv TRYON 1.8 mpi Flex    425
Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p    425
Focus 2.0 16V/SE/SE Plus Flex 5p Aut.    400
Saveiro 1.6 Mi/ 1.6 Mi Total Flex 8V     400
Corvette 5.7/ 6.0, 6.2 Targa/Stingray    375
...
STEPWAY Zen Flex 1.0 12V Mec.             2
Saveiro Robust 1.6 Total Flex 16V CD      2
Saveiro Robust 1.6 Total Flex 16V        2
Gol Last Edition 1.0 Flex 12V 5p         2
Polo Track 1.0 Flex 12V 5p               2
Name: count, Length: 2112, dtype: int64
```

```
[14]: # contagem dos modelos por marca
dados['brand'].value_counts()
```

```
[14]: brand
Fiat                44962
VW - VolksWagen    44312
GM - Chevrolet     38590
Ford               33151
Renault            29192
Nissan              12090
Name: count, dtype: int64
```

```
[16]: # contagem dos modelos por câmbio
dados['gear'].value_counts()
```

```
[16]: gear
      manual      161885
      automatic   40412
      Name: count, dtype: int64
```

```
[17]: dados['fuel'].value_counts()
```

```
[17]: fuel
      Gasoline    168685
      Diesel      30099
      Alcohol     3513
      Name: count, dtype: int64
```

- Dê um breve explicação (máximo de quatro linhas) sobre os principais resultados encontrados na Análise Exploratória dos dados

A partir da análise exploratória dos dados podemos identificar que há valores ausentes no conjunto de dados. Como os valores ausentes estão presentes em todas as colunas, decidi remover as linhas com valores ausentes.

No geral, o preço médio de um carro no Brasil custa aproximadamente R\$52,756,91. Um dos modelos mais vendidos é *Palio Week. Adv/Adv TRYON 1.8 mpi Flex*, com câmbio *manual* e motor 1.6.

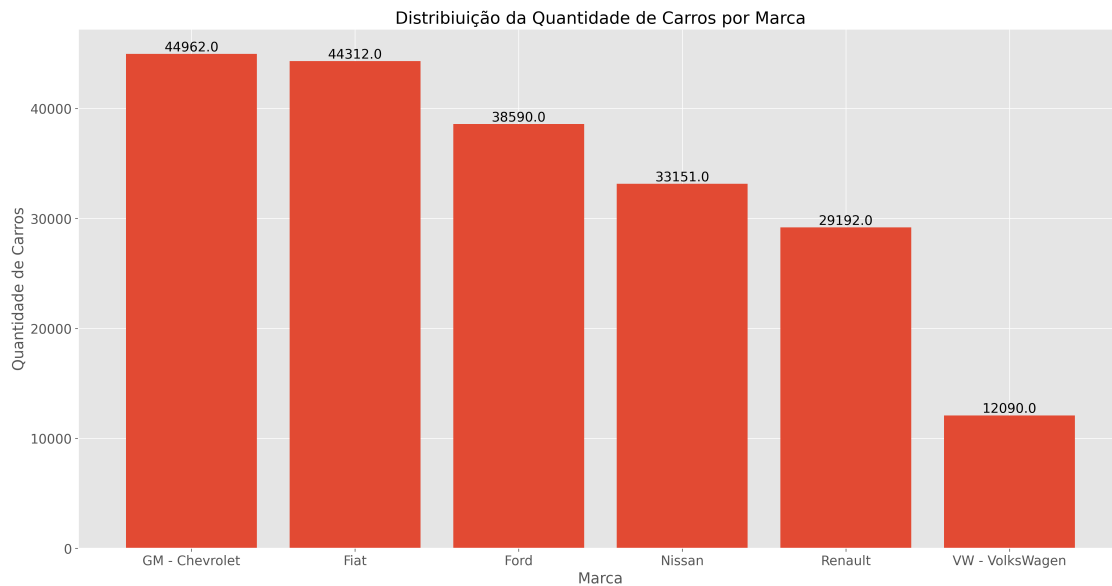
4 Visualização dos Dados

- Gere um gráfico da distribuição da quantidade de carros por marca

```
[118]: # visualizado distribuição da quantidade de carros por marca
plt.figure(figsize=(20,10));
grafico_1 = plt.bar(dados['brand'].unique(), dados['brand'].
    ↳value_counts())
plt.title("Distribuição da Quantidade de Carros por Marca", fontsize=18)
plt.ylabel("Quantidade de Carros", fontsize=16)
plt.xlabel("Marca", fontsize=16)

# aumentar o tamanho dos eixos
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)

# colocando o rótulo das barras
plt.bar_label(grafico_1, fmt="%.01f", size=14, label_type='edge');
```

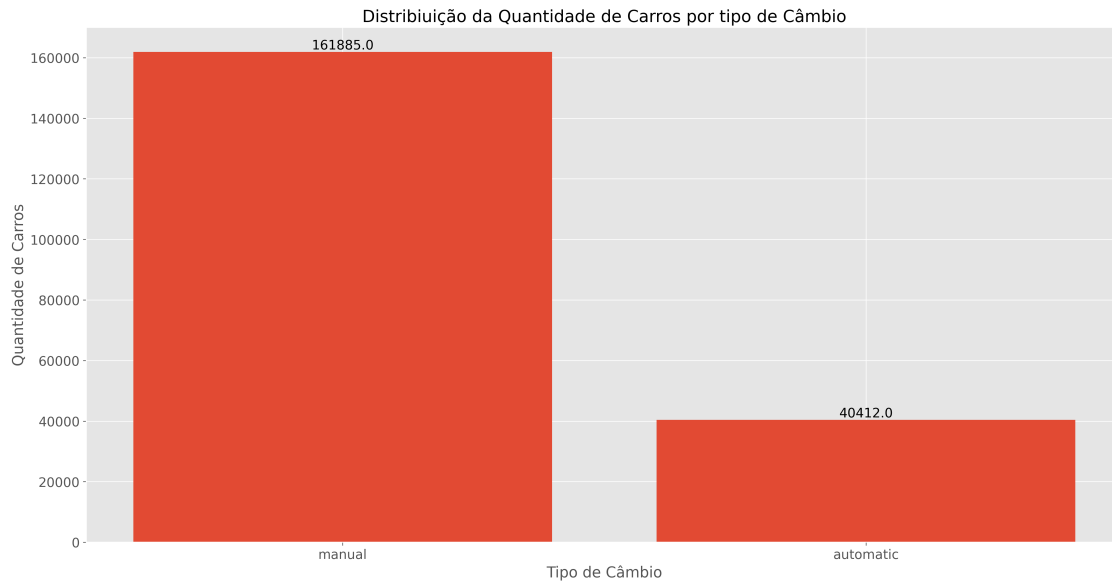


- Gere um gráfico da distribuição da quantidade de carros por tipo de engrenagem do carro

```
[119]: # visualizado distribuição da quantidade de carros por tipo de câmbio
plt.figure(figsize=(20,10));
grafico_2 = plt.bar(dados['gear'].unique(), dados['gear'].value_counts())
plt.title("Distribuição da Quantidade de Carros por tipo de Câmbio",
          →fontsize=18)
plt.ylabel("Quantidade de Carros", fontsize=16)
plt.xlabel("Tipo de Câmbio", fontsize=16)

# aumentar o tamanho dos eixos
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)

# colocando o rótulo das barras
plt.bar_label(grafico_2, fmt="%.01f", size=14, label_type='edge');
```



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

```
[21]: # ordenando coluna month_of_reference
dados['month_of_reference'] = pd.Categorical(dados['month_of_reference'],
                                             categories=dados['month_of_reference'].unique(),
                                             ordered=True)
dados.head()
```

```
[21]:   year_of_reference month_of_reference fiipe_code authentication \
0          2021.0          January  004001-0  cfzltzfwrcp
1          2021.0          January  004001-0  cdqwxpw3y2p
2          2021.0          January  004001-0  cb1t3xwwj1xp
3          2021.0          January  004001-0  cb9gct6j65r0
4          2021.0          January  004003-7  g15wg0gbz1fx

      brand          model      fuel
↳ gear \
0  GM - Chevrolet  Corsa Wind 1.0 MPFI / EFI 2p  Gasoline
↳ manual
1  GM - Chevrolet  Corsa Wind 1.0 MPFI / EFI 2p  Gasoline
↳ manual
2  GM - Chevrolet  Corsa Wind 1.0 MPFI / EFI 2p  Gasoline
↳ manual
```



```

3  GM - Chevrolet          Corsa Wind 1.0 MPFI / EFI 2p   Alcohol  ↵
   ↪manual
4  GM - Chevrolet  Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI   Gasoline  ↵
   ↪manual

```

	engine_size	year_model	avg_price_brl
0	1	2002.0	9162.0
1	1	2001.0	8832.0
2	1	2000.0	8388.0
3	1	2000.0	8453.0
4	1,6	2001.0	12525.0

```

[22]: # coletando dados de 2022
dados_2022 = dados[dados['year_of_reference'] == 2022]
dados_2022.shape

```

```
[22]: (97798, 11)
```

```

[23]: # calculando a média mensal do preço dos carros
media_precos_2022 = dados_2022.
    ↪groupby(['month_of_reference'])['avg_price_brl'].mean()
media_precos_2022.head(12)

```

```

[23]: month_of_reference
January      54840.270037
February     55824.519882
March        56848.951914
April        57150.037325
May          57799.763776
June         58065.611398
July         57893.997056
August       57923.544105
September    58198.936989
October      58227.410144
November     58215.626236
December     57998.054038
Name: avg_price_brl, dtype: float64

```

```

[24]: # resetando index
media_precos_2022 = media_precos_2022.reset_index(name='avg_price_month')
media_precos_2022.head()

```

```

[24]:  month_of_reference  avg_price_month
0           January      54840.270037

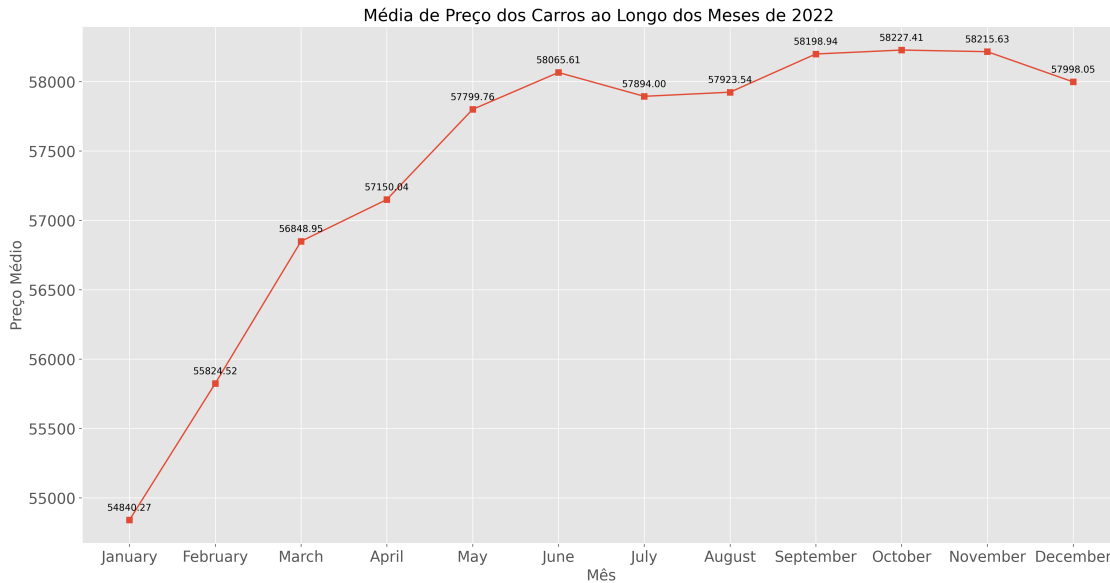
```

1	February	55824.519882
2	March	56848.951914
3	April	57150.037325
4	May	57799.763776

```
[120]: # visualizado o preço médio dos carros ao longo dos meses de 2022
plt.figure(figsize=(20,10));
grafico_1 = plt.plot(media_precos_2022['month_of_reference'],
    ↳media_precos_2022['avg_price_month'], marker='s')
plt.title("Média de Preço dos Carros ao Longo dos Meses de 2022 ",
    ↳fontsize=18)
plt.ylabel("Preço Médio", fontsize=16)
plt.xlabel("Mês", fontsize=16)

# adiciona anotações de cada ponto ao gráfico
for x, y in zip(media_precos_2022['month_of_reference'],
    ↳media_precos_2022['avg_price_month']):
    label = "{:.2f}".format(y)
    plt.annotate(
        label,
        (x,y),
        textcoords="offset points",
        xytext=(0,10),
        ha='center'
    )

# aumentar o tamanho dos eixos
plt.xticks(fontsize=16)
plt.yticks(fontsize=16);
```



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

```
[26]: preco_marca_cambio = dados.groupby(['brand', 'gear'])['avg_price_brl'].
      ↪mean()
      preco_marca_cambio.head()
```

```
[26]: brand      gear
      Fiat      automatic    97396.801936
           manual      39694.442749
      Ford      automatic    84769.106720
           manual      51784.851550
      GM - Chevrolet automatic    88156.919439
      Name: avg_price_brl, dtype: float64
```

```
[27]: # resetando index
      preco_marca_cambio = preco_marca_cambio.
      ↪reset_index(name='avg_price_brand_gear')
      preco_marca_cambio.head()
```

```
[27]:      brand      gear  avg_price_brand_gear
0      Fiat  automatic    97396.801936
1      Fiat   manual     39694.442749
2      Ford  automatic    84769.106720
3      Ford   manual     51784.851550
4  GM - Chevrolet  automatic    88156.919439
```

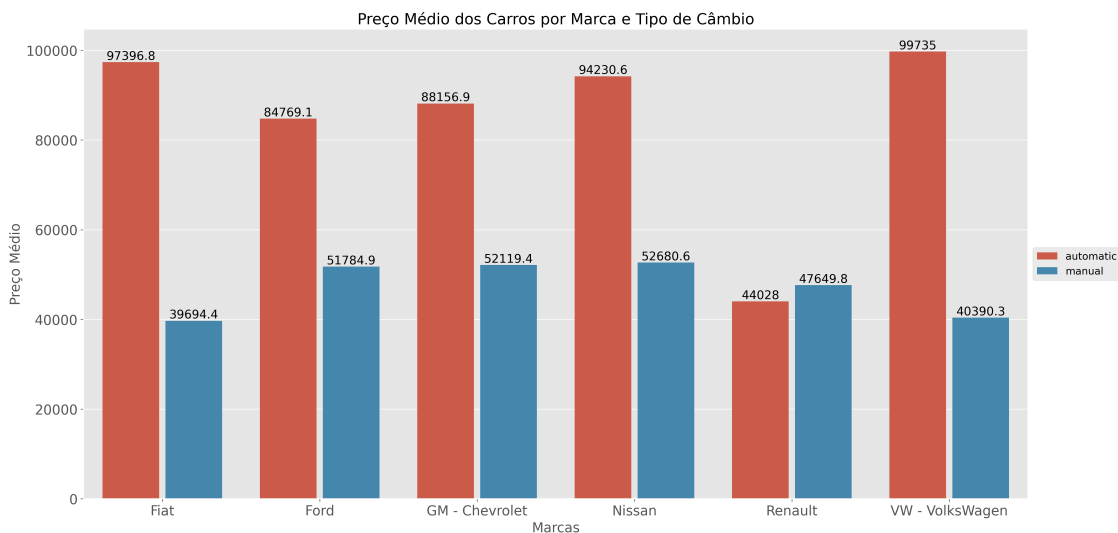
```
[121]: # visualizando a distribuição média de preço dos carros por marca e tipo
        ↳ de câmbio
plt.figure(figsize=(20,10))
ax = sns.barplot(x="brand", y="avg_price_brand_gear", hue="gear",
                 data=preco_marca_cambio,
        ↳ hue_order=preco_marca_cambio['gear'].unique(),
                 gap=0.1);

ax.legend(loc='center left', bbox_to_anchor=(1, 0.5), ncol=1, fontsize=12)

ax.bar_label(ax.containers[0], fontsize=14);
ax.bar_label(ax.containers[1], fontsize=14);

plt.title("Preço Médio dos Carros por Marca e Tipo de Câmbio",
        ↳ fontsize=18)
# renomeando os eixos
plt.ylabel("Preço Médio", fontsize=16)
plt.xlabel("Marcas", fontsize=16)

# aumenta tamanho fonte dos eixos
plt.xticks(fontsize=16)
plt.yticks(fontsize=16);
```



- Dê uma breve explicação (máximo de quatro linhas) sobre os resultados gerados no item d

O preço médio do carro com câmbio automático é relativamente mais alto quando comparado aos de câmbio manual para todas as marcas, com exceção da Renault. VolksWagen e Fiat

lideram as marcas que possuem maior preço médio para carros de câmbio automático, ao passo que lideram também as marcas com menor preço médio dos carros com câmbio manual.

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

```
[29]: # distribuição da média de preço dos carros por marca e tipo de
      ↳ combustível
preco_marca_combustivel = dados.
      ↳ groupby(['brand', 'fuel'])['avg_price_brl'].mean()
preco_marca_combustivel.head()
```

```
[29]: brand  fuel
      Fiat  Alcohol    11509.514419
           Diesel     99814.451429
           Gasoline   37197.294483
      Ford  Alcohol    10148.906667
           Diesel     94522.454826
      Name: avg_price_brl, dtype: float64
```

```
[30]: # resetando index
preco_marca_combustivel = preco_marca_combustivel.
      ↳ reset_index(name="avg_price_brand_fuel")
preco_marca_combustivel.head()
```

```
[30]:   brand    fuel  avg_price_brand_fuel
0  Fiat  Alcohol    11509.514419
1  Fiat   Diesel    99814.451429
2  Fiat Gasoline   37197.294483
3  Ford  Alcohol    10148.906667
4  Ford   Diesel    94522.454826
```

```
[122]: # visualizando a distribuição da média de preço dos carros por marca e
      ↳ tipo de combustível

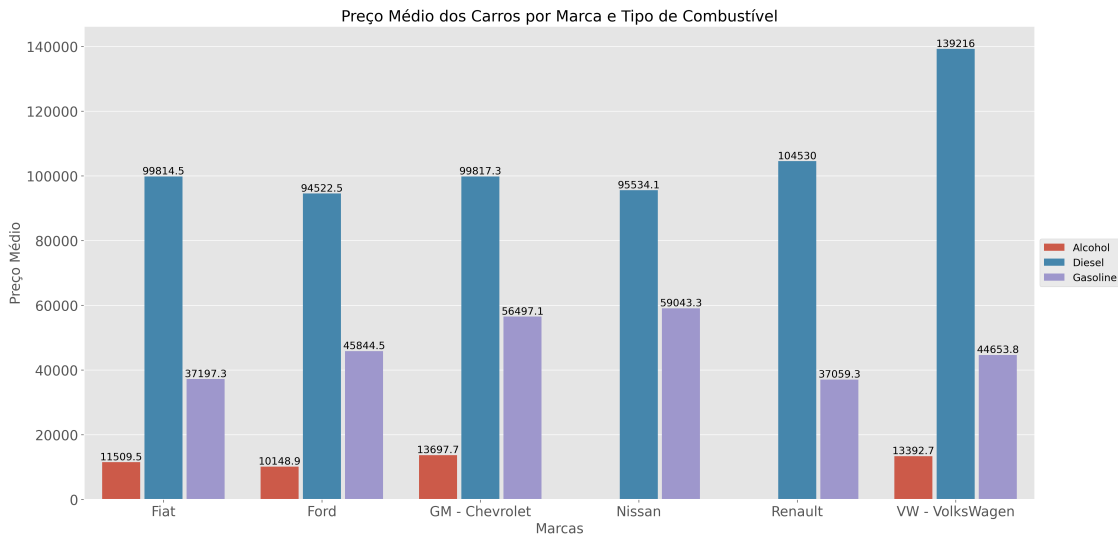
plt.figure(figsize=(20,10))
ax = sns.barplot(x="brand", y="avg_price_brand_fuel", hue="fuel",
                 data=preco_marca_combustivel,
      ↳ hue_order=preco_marca_combustivel['fuel'].unique(),
                 gap=0.1);

ax.legend(loc='center left', bbox_to_anchor=(1, 0.5), ncol=1, fontsize=12)

for c in range(len(ax.containers)):
    ax.bar_label(ax.containers[c], fontsize=12);
```

```
plt.title("Preço Médio dos Carros por Marca e Tipo de Combustível",
         ↪fontsize=18)
# renomeando os eixos
plt.ylabel("Preço Médio", fontsize=16)
plt.xlabel("Marcas", fontsize=16)

# aumenta tamanho fonte dos eixos
plt.xticks(fontsize=16)
plt.yticks(fontsize=16);
```



- Dê uma breve explicação (máximo de quatro linhas) sobre os resultados gerados no item f
1. O preço médio do carro à diesel é muito mais elevado para todas as marcas, sendo a VolksWagen a marca com maior preço médio para esta categoria.
 2. O preço médio do carro à álcool lidera com os menores valores.
 3. Os carros à gasolina com menor preço médio são das marcas Fiat e Renault.

5 Predição do Preço Médio dos Carros com Machine Learning

- Escolha as variáveis numéricas (modelos de Regressão) para serem as variáveis independentes do modelo. A variável target é avg_price. Observação: caso julgue necessário, faça a transformação de variáveis categóricas em variáveis numéricas para inputar no

modelo. Indique quais variáveis foram transformadas e como foram transformadas.

```
[32]: dados.columns
```

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

```
[33]: print(dados.dtypes)
```

```
year_of_reference    float64  
month_of_reference   category  
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
```

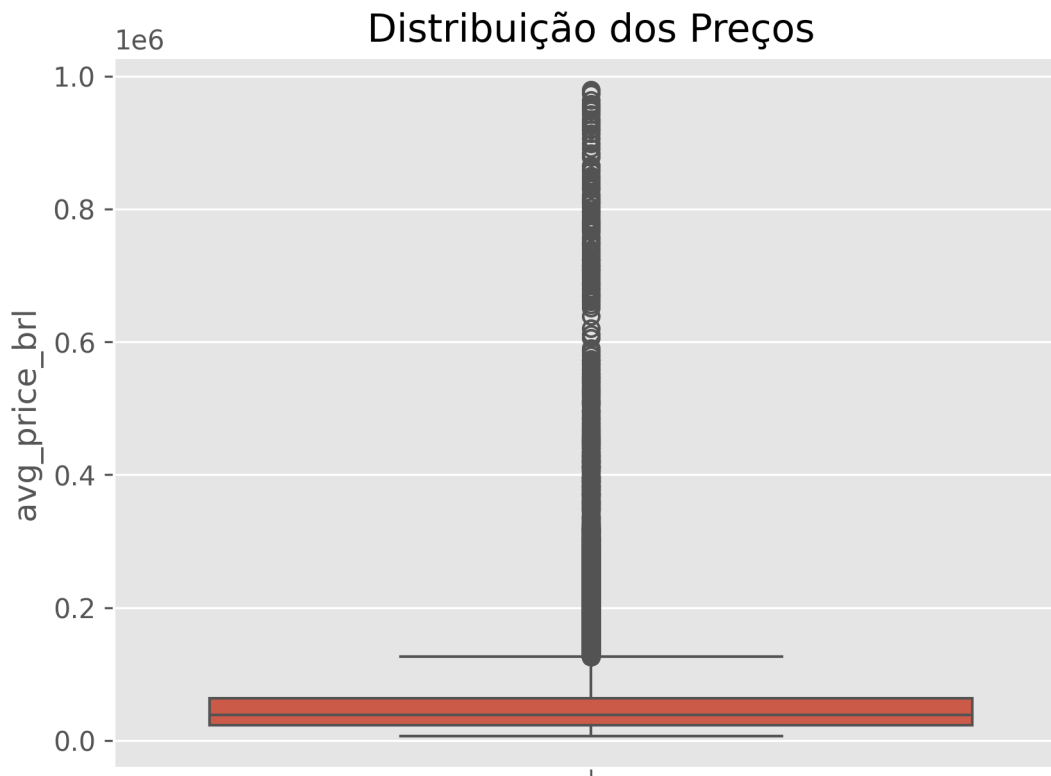
As variáveis independentes selecionadas e que, a princípio, podem ser importantes para predição foram: 1. 'brand' 2. 'model' 3. 'fuel' 4. 'gear' 5. 'engine_size' 6. 'year_model'

A variável dependente ou target é: 1. 'avg_price_brl'

- Visualizando a distribuição da variável dependente.

```
[123]: # analisando a variável target  
sns.boxplot(dados['avg_price_brl']).set_title("Distribuição dos Preços")
```

```
[123]: Text(0.5, 1.0, 'Distribuição dos Preços')
```



O boxplot evidencia a presença de muitos outliers, no entanto, nenhuma forma de tratamento destes outliers será aplicada para este trabalho.

5.1 Pré-processamento dos dados

```
[76]: # selecionando variáveis de interesse
dados_vi = dados[['brand', 'model', 'fuel', 'gear', 'engine_size',
→ 'year_model', 'avg_price_brl']]
dados_vi
```

```
[76]:
```

	brand	model	fuel
0	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline
1	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline
2	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline
3	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Alcohol
4	GM - Chevrolet	Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI	Gasoline
...
202292	VW - Volkswagen	Saveiro Robust 1.6 Total Flex 16V	Gasoline
202293	VW - Volkswagen	Gol Last Edition 1.0 Flex 12V 5p	Gasoline

202294	VW - Volkswagen	Gol Last Edition 1.0 Flex 12V 5p	Gasoline
202295	VW - Volkswagen	Polo Track 1.0 Flex 12V 5p	Gasoline
202296	VW - Volkswagen	Polo Track 1.0 Flex 12V 5p	Gasoline

	gear	engine_size	year_model	avg_price_brl
0	manual	1	2002.0	9162.0
1	manual	1	2001.0	8832.0
2	manual	1	2000.0	8388.0
3	manual	1	2000.0	8453.0
4	manual	1,6	2001.0	12525.0
...
202292	manual	1,6	2023.0	86038.0
202293	manual	1	2023.0	95997.0
202294	manual	1	2023.0	87828.0
202295	manual	1	2023.0	80845.0
202296	manual	1	2023.0	74458.0

[202297 rows x 7 columns]

```
[77]: # transformando engine_size em variável contínua.
dados_vi['engine_size'] = dados_vi['engine_size'].apply(lambda x: x.
    →replace(',', '.')) # substitui a vírgula pelo ponto em valores decimais
dados_vi['engine_size'] = dados_vi['engine_size'].astype(float)
    → # transforma dados de str para float
dados_vi
```

```
[77]:
```

	brand	model	fuel
0	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline
1	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline
2	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline
3	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Alcohol
4	GM - Chevrolet	Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI	Gasoline
...
202292	VW - Volkswagen	Saveiro Robust 1.6 Total Flex 16V	Gasoline
202293	VW - Volkswagen	Gol Last Edition 1.0 Flex 12V 5p	Gasoline
202294	VW - Volkswagen	Gol Last Edition 1.0 Flex 12V 5p	Gasoline
202295	VW - Volkswagen	Polo Track 1.0 Flex 12V 5p	Gasoline
202296	VW - Volkswagen	Polo Track 1.0 Flex 12V 5p	Gasoline

	gear	engine_size	year_model	avg_price_brl
0	manual	1.0	2002.0	9162.0
1	manual	1.0	2001.0	8832.0
2	manual	1.0	2000.0	8388.0

3	manual	1.0	2000.0	8453.0
4	manual	1.6	2001.0	12525.0
...
202292	manual	1.6	2023.0	86038.0
202293	manual	1.0	2023.0	95997.0
202294	manual	1.0	2023.0	87828.0
202295	manual	1.0	2023.0	80845.0
202296	manual	1.0	2023.0	74458.0

[202297 rows x 7 columns]

As variáveis categóricas *brand*, *model*, *fuel* e *gear* serão transformadas em variáveis numéricas por meio da técnica label encoding.

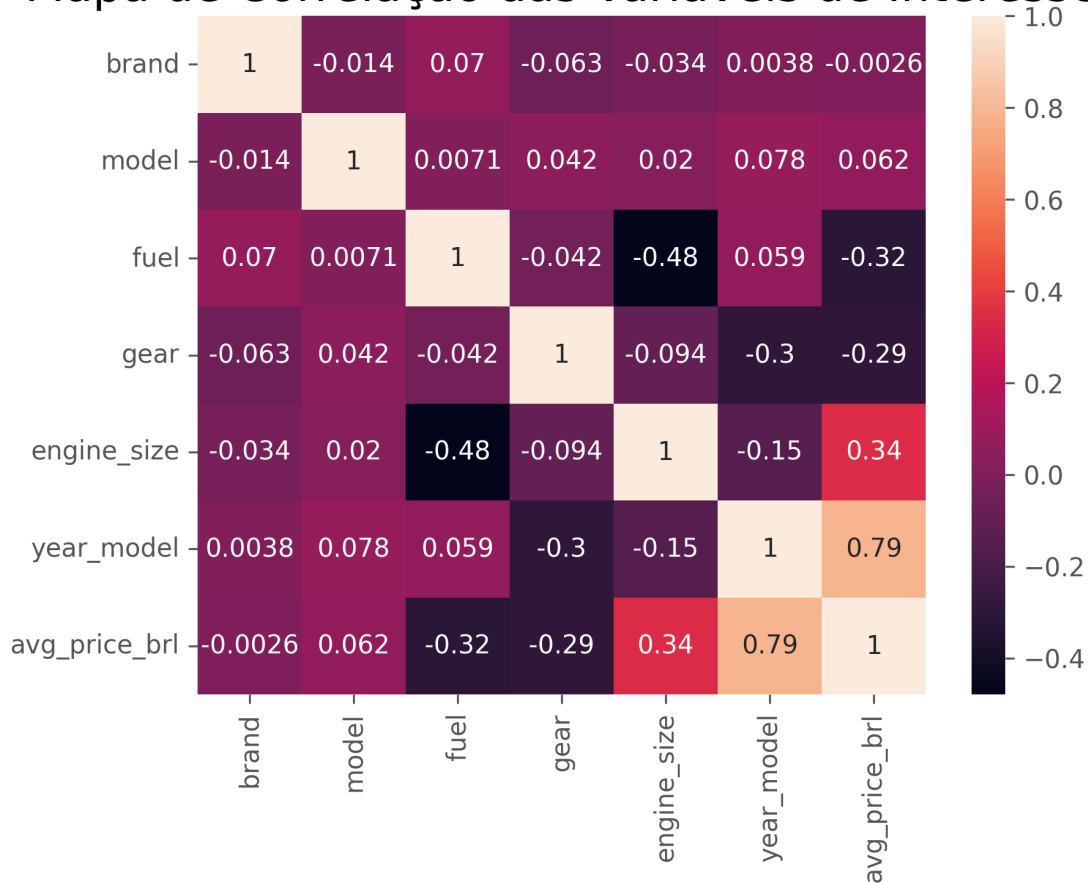
```
[78]: # transformando as variáveis categóricas
le = LabelEncoder()
columns = ['brand', 'model', 'fuel', 'gear']
for column in columns:
    dados_vi[column] = le.fit_transform(dados_vi[column])

dados_vi.head()
```

```
[78]:   brand  model  fuel  gear  engine_size  year_model  avg_price_brl
0      2    297     2     1           1.0      2002.0           9162.0
1      2    297     2     1           1.0      2001.0           8832.0
2      2    297     2     1           1.0      2000.0           8388.0
3      2    297     0     1           1.0      2000.0           8453.0
4      2    260     2     1           1.6      2001.0          12525.0
```

```
[124]: # Mapa de correlação das variáveis com a variável objetivo
sns.heatmap(dados_vi.corr('spearman'), annot=True)
plt.title("Mapa de Correlação das Variáveis de Interesse", fontsize=18)
plt.show()
```

Mapa de Correlação das Variáveis de Interesse



O mapa de correlação mostra uma correlação positiva com a variável target (`avg_price_brl`) mais evidente entre as variáveis `engine_size`, `year_model`. Além disso, uma correlação negativa com a variável target está mais acentuada entre as variáveis `fuel` e `gear`. As variáveis `brand` e `model` apresentam correlação muito baixas e podem ser candidatas a serem desconsideradas, caso necessário, em modelagens futuras.

5.2 Separando os conjuntos de treinamento e testes

```
[82]: # conjunto de variáveis dependentes
X = dados_vi.drop(['avg_price_brl'], axis=1)
X
```

```
[82]:
```

	brand	model	fuel	gear	engine_size	year_model
0	2	297	2	1	1.0	2002.0
1	2	297	2	1	1.0	2001.0
2	2	297	2	1	1.0	2000.0

3	2	297	0	1	1.0	2000.0
4	2	260	2	1	1.6	2001.0
...
202292	5	1678	2	1	1.6	2023.0
202293	5	720	2	1	1.0	2023.0
202294	5	720	2	1	1.0	2023.0
202295	5	1374	2	1	1.0	2023.0
202296	5	1374	2	1	1.0	2023.0

[202297 rows x 6 columns]

```
[83]: # conjunto da variável independente
Y = dados_vi['avg_price_brl']
Y
```

```
[83]: 0          9162.0
      1          8832.0
      2          8388.0
      3          8453.0
      4         12525.0
      ...
      202292      86038.0
      202293      95997.0
      202294      87828.0
      202295      80845.0
      202296      74458.0
      Name: avg_price_brl, Length: 202297, dtype: float64
```

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

```
[84]: # separando conjuntos de treinamento e teste
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25,
→random_state=74)
```

```
[85]: print(X_train.shape)
      X_train.head()
```

(151722, 6)

```
[85]:      brand  model  fuel  gear  engine_size  year_model
      181406      1   1430      1      1          2.8      2003.0
      72297      2     50      2      0          2.0      2003.0
      91746      1    556      2      1          2.0      2008.0
      193272     5   1315      2      1          3.6      2009.0
      73555      0   1985      2      1          1.5      2000.0
```

```
[86]: print(X_test.shape)
      X_test.head()
```

(50575, 6)

```
[86]:      brand  model  fuel  gear  engine_size  year_model
      197874     1   560    2     1           1.8      2003.0
      191045     3   643    1     0           2.3      2019.0
      158570     4   194    2     0           1.0      2005.0
      9687      0   383    1     1           2.8      2001.0
      141579     3   646    1     1           2.8      2004.0
```

```
[88]: print(Y_train.shape)
      Y_train
```

(151722,)

```
[88]: 181406      34073.0
      72297      17397.0
      91746      23701.0
      193272     63339.0
      73555       7626.0
      ...
      195716     12926.0
      11070     140995.0
      190665     53563.0
      7305      90030.0
      13147       9361.0
      Name: avg_price_brl, Length: 151722, dtype: float64
```

5.3 Treinamento dos Modelos

- Treine modelos RandomForest (biblioteca RandomForestRegressor) e XGBoost (biblioteca XGBRegressor) para predição dos preços dos carros. Observação: caso julgue necessário, mude os parâmetros dos modelos e rode novos modelos. Indique quais parâmetros foram inputados e indique o treinamento de cada modelo.

5.3.1 Random Forest

```
[90]: # random forest sem especificação dos parâmetros
      modelo_rf = RandomForestRegressor()
```

```
[91]: # ajuste do modelo
      modelo_rf.fit(X_train, Y_train)
```

```
[91]: RandomForestRegressor()
```

- Grave os valores preditos em variáveis criadas.

```
[110]: # predição com o conjunto de teste
rf_pred = modelo_rf.predict(X_test)
```

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

```
[97]: # análise de importância das variáveis
importancias = modelo_rf.feature_importances_
feature_importances = pd.DataFrame(importancias, index=X_train.columns,
→columns=['importance']).sort_values('importance', ascending=False)
feature_importances
```

```
[97]:
```

	importance
engine_size	0.454362
year_model	0.400901
model	0.057540
fuel	0.035217
gear	0.033895
brand	0.018085

- Dê uma breve explicação (máximo de quatro linhas) sobre os resultados encontrados na análise de importância de variáveis

A análise de importância mostra que *engine_size* e *year_model* foram as variáveis mais importantes para o modelo *Random Forest*. As demais variáveis possuem grau de importância muito baixos para o desempenho geral do modelo.

Mettricas de Acurácia

```
[111]: mse = mean_squared_error(Y_test, rf_pred)
mae = mean_absolute_error(Y_test, rf_pred)
r2 = r2_score(Y_test, rf_pred)
print(f'MSE: {mse} \nMAE: {mae} \nR2: {r2}')
```

```
MSE: 53141095.177221134
MAE: 4204.109339157372
R2: 0.9807983796529883
```

5.3.2 XGBoost

```
[101]: modelo_xgb = XGBRegressor()
```

```
[102]: # ajuste do modelo
modelo_xgb.fit(X_train, Y_train)
```

```
[102]: 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,
                    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,
                    ↪monotone_constraints=None,
                    multi_strategy=None, n_estimators=None, n_jobs=None,
                    num_parallel_tree=None, random_state=None, ...)
```

- Grave os valores preditos em variáveis criadas.

```
[103]: # predição com o conjunto de teste
xgb_pred = modelo_xgb.predict(X_test)
```

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

```
[104]: importancias_xgb = modelo_xgb.feature_importances_
feature_importances = pd.DataFrame(importancias_xgb, index=X_train.
    ↪columns, columns=['importance']).sort_values('importance',
    ↪ascending=False)
feature_importances
```

```
[104]:
```

	importance
engine_size	0.481119
year_model	0.173308
fuel	0.150183
gear	0.119191
brand	0.056852
model	0.019347

- Dê uma breve explicação (máximo de quatro linhas) sobre os resultados encontrados na análise de importância de variáveis

A análise de importância mostra que *engine_size* foi a variável mais importante para o modelo *XGBoost*. O modelo *XGBoost* diminuiu a importância de *year_model*, ao passo que aumentou a importância de *fuel* e *gear*. As demais variáveis possuem grau de importância muito baixos para o desempenho geral do modelo.

```
[108]: mse = mean_squared_error(Y_test, xgb_pred)
mae = mean_absolute_error(Y_test, xgb_pred)
r2 = r2_score(Y_test, xgb_pred)
print(f'MSE: {mse} \nMAE: {mae}\nR2: {r2}')
```

MSE: 70391500.07536449

MAE: 4913.773608589116

R2: 0.9745652426696096

6 Conclusão

- Escolha o melhor modelo com base nas métricas de avaliação MSE, MAE e R2.
- Dê uma breve explicação (máximo de quatro linhas) sobre qual modelo gerou o melhor resultado e a métrica de avaliação utilizada.

Após análise das métricas de desempenho dos modelos *Random Forest* e *XGBoost* podemos observar que ambos obtiveram bom desempenho na predição para o conjunto de testes quando observa-se o valor de R^2 , sendo:

- $R^2 = 0.98$ para *Random Forest*, e
- $R^2 = 0.97$ para *XGBoost*.

Mesmo com uma diferença muito pequena, *Random Forest* obteve melhor desempenho.