

1 a)

✓ 1 - Análise Exploratória dos dados

A partir da base de dados precos_carros_brasil.csv, execute as seguintes tarefas

✓ a. Carregue a base de dados media_precos_carros_brasil.csv

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.impute import SimpleImputer
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

#Carregando a base de dados.

dados = pd.read_csv('sample_data/precos_carros_brasil.csv')
```

1 b)

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

```
[2] # Verificando a existencia de elementos sem valor nenhum

dados.isna().sum()

# Metodo para remover linhas que contenham linhas com valores faltantes

# A função dropna() apaga as linhas que contem entradas com dados "não disponiveis"
dados = dados.dropna()
```


```
dados.head()
```

	year_of_reference	month_of_reference	fipecode	authentication	brand	model	fuel	gear	engine_size	year_model	avg_price_brl
0	2021.0	January	004001-0	cfzictzfwrcp	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/Chamo 1.6 MPFI / EFI	Gasoline	manual	1,6	2001.0	12525.0

1 c)

✓ c. Verifique se há dados duplicados nos dados

✓
0s

 dados.duplicated().sum()

2

1 d)

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



✓
0s

```
# Criando duas categorias para armazenar as colunas categoricas e numéricas

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']
```

✓
0s

```
# Imprimindo o resumo de informações das variáveis numéricas
dados[numericas_cols].describe()
```

	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	

✓
0s

```
[7] # Imprimindo o resumo de informações das variáveis categóricas
dados[categoricas_cols].describe()
```

	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size	
count	202297	202297	202297	202297	202297	202297	202297	202297	
unique	12	2091	202295	6	2112	3	2	29	
top	January	003281-6	3r6c277cnqcb	Fiat	Pallo Week. Adv/Adv TRYON 1.8 mpi Flex	Gasoline	manual	1,6	
freq	24260	425	2	44962	425	168685	161885	47420	

1 e)

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

```
✓ [8] # Imprimindo a contagem de valores - Modelo  
0s dados['model'].value_counts()
```

```
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: model, Length: 2112, dtype: int64
```

```
✓ [9] # Imprimindo a contagem de valores - Marca  
0s dados['brand'].value_counts()
```

```
Fiat                44962  
VW - Volkswagen    44312  
GM - Chevrolet     38590  
Ford               33151  
Renault            29192  
Nissan              12090  
Name: brand, dtype: int64
```

1 f) Dentro os resultados obtidos na Análise Exploratória, podemos dizer que a base de dados original possui 11 colunas e 202297 dados, destes, 65245 são valores que possuem dados faltantes. Depois do tratamento para eliminar valores faltantes obtivemos uma quantidade total de 137052 registros. Podemos constatar também que após os tratamentos, há 2 dados duplicados. O modelo de carro que aparece com mais frequência é um empate entre o "Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p" e o "Palio Week. Adv/Adv TRYON 1.8 mpi Flex" que aparecem ambos 425 vezes, ao passo que a marca que mais aparece é a Fiat com 44962 ocorrências.

2 a)

✓ 2 Visualização dos dados

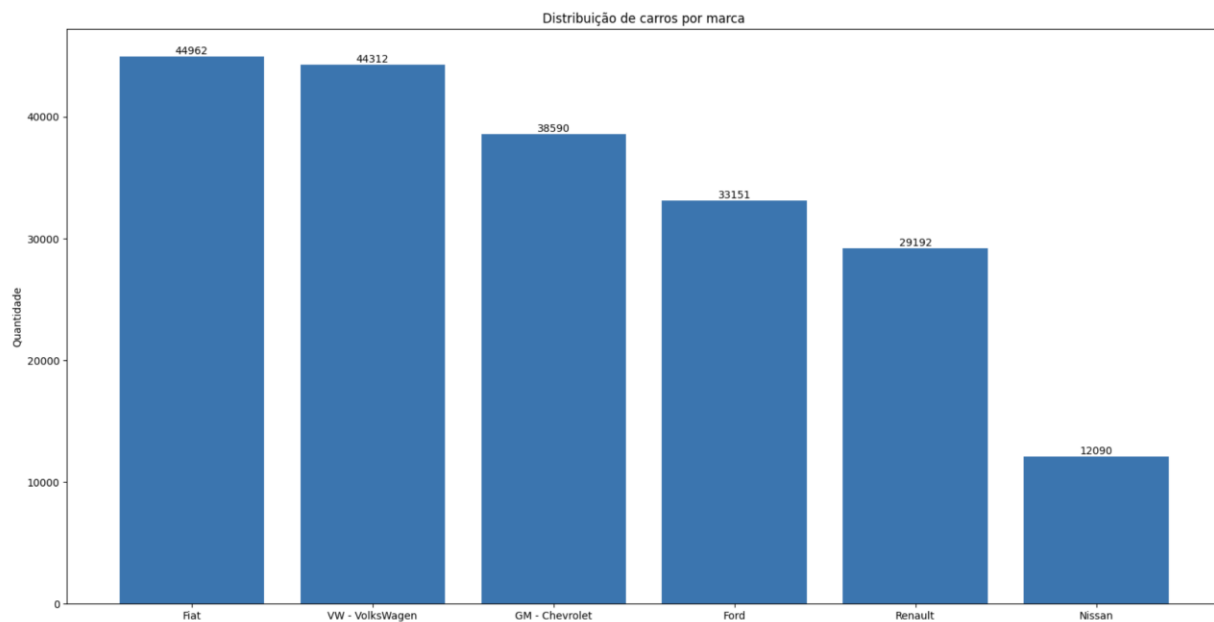
A partir da base de dados `precos_carros_brasil.csv`, execute as seguintes tarefas:

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

```
3 valores_contados = dados['brand'].value_counts()

plt.figure(figsize=(20,10))
grafico_1 = plt.bar(valores_contados.index, valores_contados.values)
plt.title('Distribuição de carros por marca')
plt.ylabel('Quantidade')
plt.bar_label(grafico_1, size=10)
```

```
4 [Text(0, 0, '44962'),
  Text(0, 0, '44312'),
  Text(0, 0, '38590'),
  Text(0, 0, '33151'),
  Text(0, 0, '29192'),
  Text(0, 0, '12090')]
```



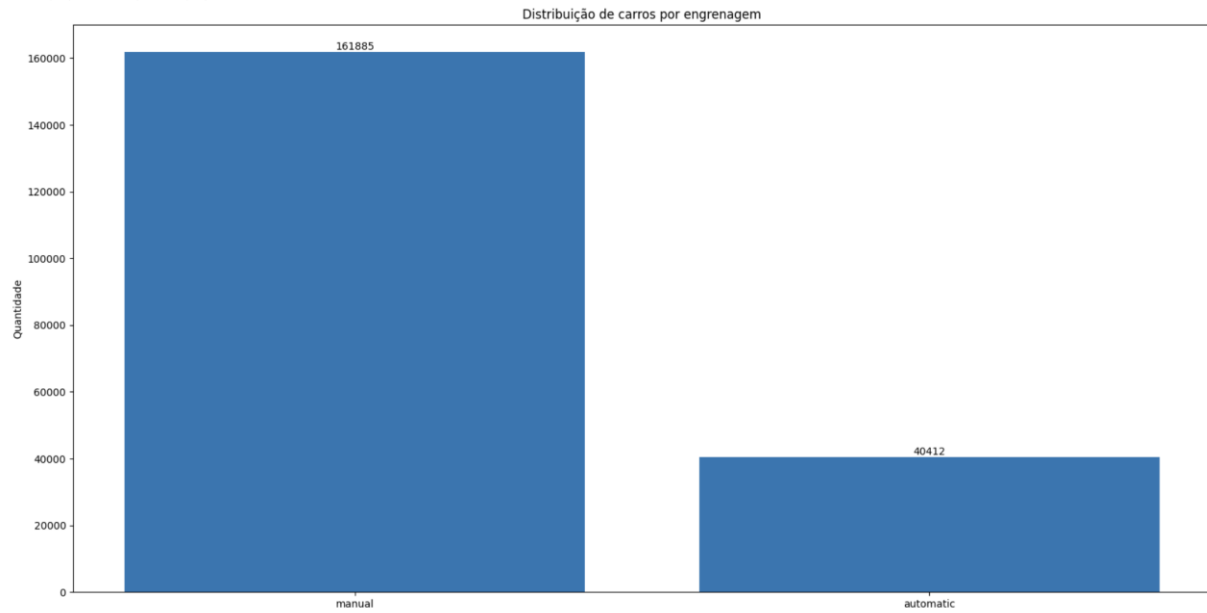
2 b)

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

```
valores_contados_2 = dados['gear'].value_counts()

plt.figure(figsize=(20,10))
grafico_2 = plt.bar(valores_contados_2.index, valores_contados_2.values)
plt.title('Distribuição de carros por engrenagem')
plt.ylabel('Quantidade')
plt.bar_label(grafico_2, size=10)
```

```
[Text(0, 0, '161885'), Text(0, 0, '40412')]
```



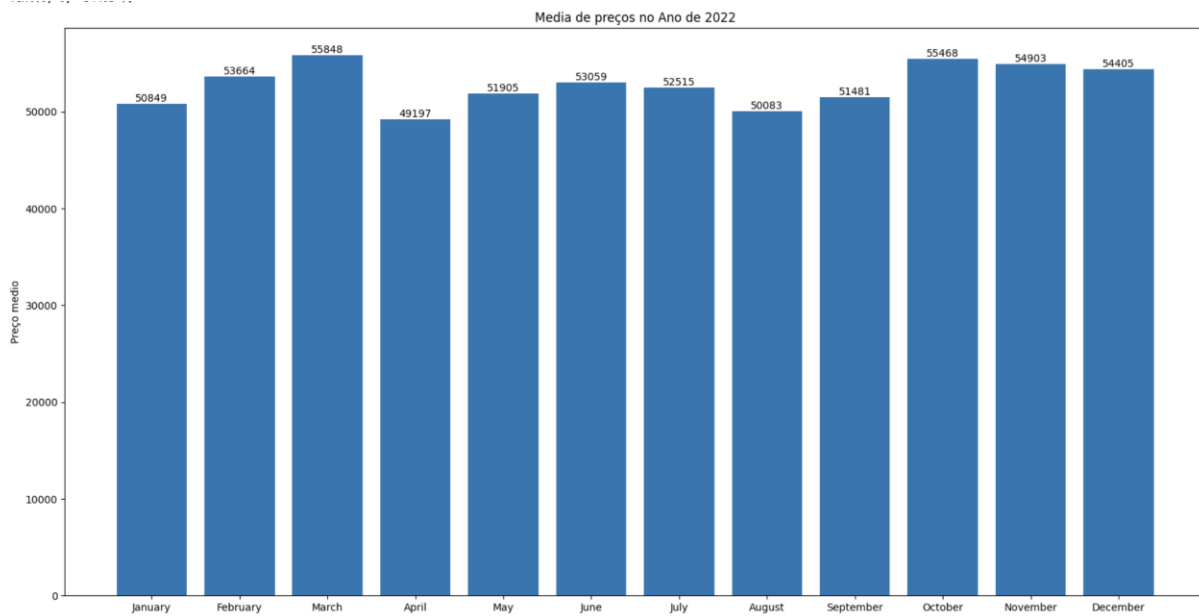
2 c)

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

```
✓ 15 ▶ ano_2022 = dados[dados.year_of_reference == 2022]

plt.figure(figsize=(20,10))
grafico_3 = plt.bar(ano_2022['month_of_reference'].unique(), dados.groupby('month_of_reference')['avg_price_brl'].mean().round())
plt.ylabel('Preço medio')
plt.title('Média de preços no Ano de 2022')
plt.bar_label(grafico_3, size=10)
```

```
➡ Text(0, 0, '49197'),
Text(0, 0, '51905'),
Text(0, 0, '53059'),
Text(0, 0, '52515'),
Text(0, 0, '50083'),
Text(0, 0, '51481'),
Text(0, 0, '55468'),
Text(0, 0, '54903'),
Text(0, 0, '54405')]
```



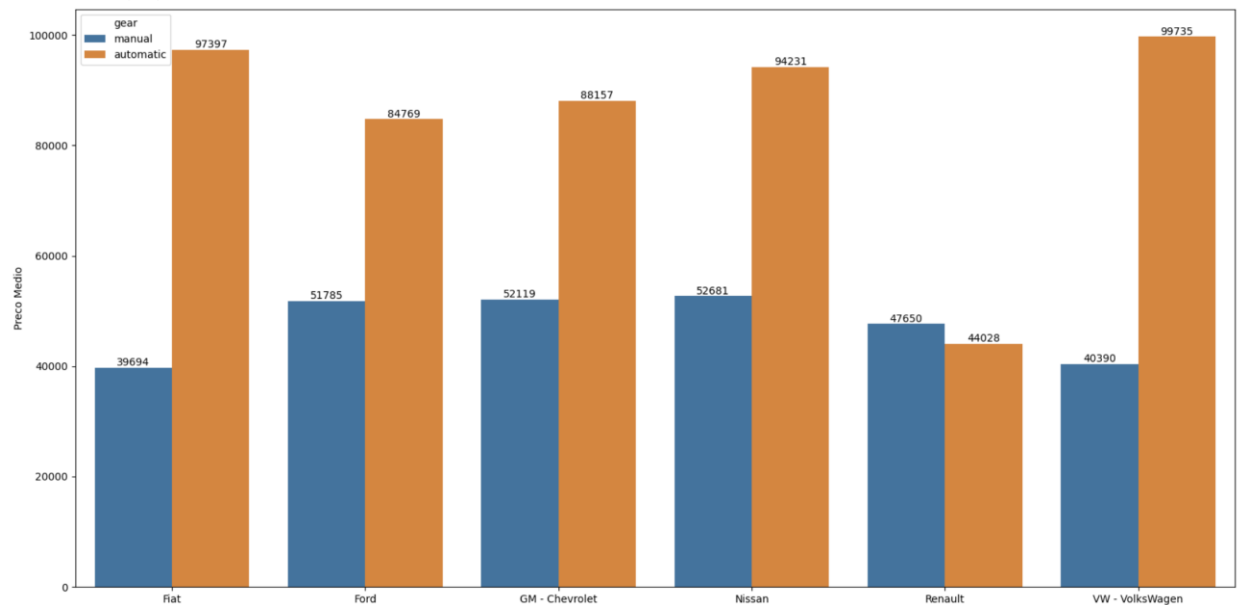
2 d)

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

```
media_marca_engrenagem = dados.groupby(['brand', 'gear'])['avg_price_brl'].mean().round()
media_marca_engrenagem = media_marca_engrenagem.reset_index(name='Preco Medio')

plt.figure(figsize=(20,10))
ax = sns.barplot(data=media_marca_engrenagem, x='brand', y='Preco Medio', hue='gear', hue_order=['manual', 'automatic'])
ax.bar_label(ax.containers[0], size=10)
ax.bar_label(ax.containers[1], size=10)
```

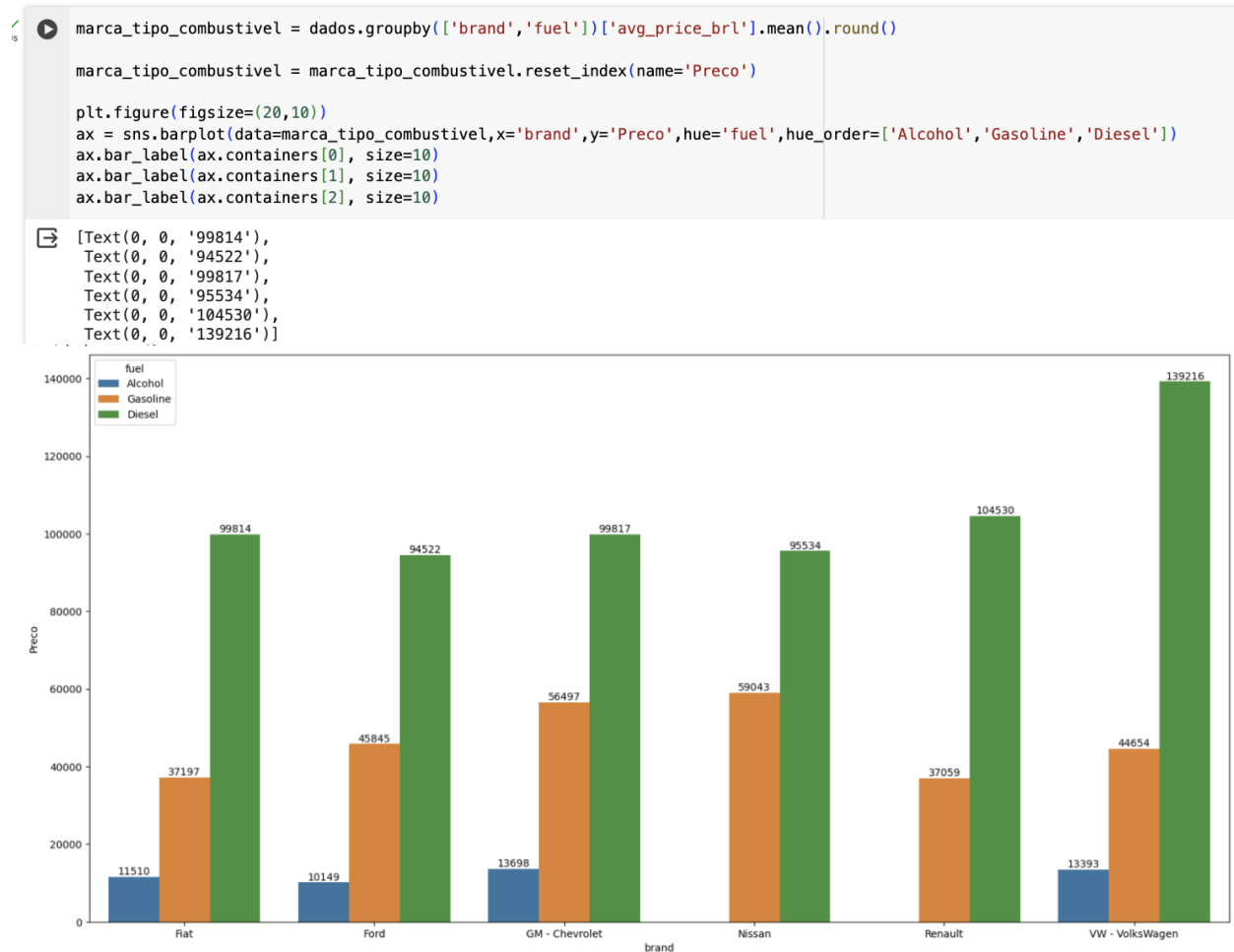
```
[Text(0, 0, '97397'),
Text(0, 0, '84769'),
Text(0, 0, '88157'),
Text(0, 0, '94231'),
Text(0, 0, '44028'),
Text(0, 0, '99735')]
```



2 e) No gráfico gerado no item d podemos verificar que todas as marcas oferecem tanto modelos de carros com câmbios (engrenagens) manuais como automáticos. Podemos também concluir que os carros com câmbio automáticos tem um preço muito maior que o de carros com engrenagens manuais, em alguns casos, como o exemplo da Fiat e da Volkswagen, essa média de preço do carro automático para o manual, pode chegar a ser o dobro do valor.

2 f)

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



2 g) O que o gráfico gerado no item f pode nos informar é que o valor dos carros que utilizam o Diesel como combustível, em media, são os mais caros. Aquele que possui o segundo valor médio seriam os carros a gasolina, com os carros a alcool no terceiro e ultimo lugar, sendo que existem marcas, como o caso da Nissan e da Renault, que de acordo com a base de dados nem chega a fornecer uma opção de carro a alcool.

3 a)

a. 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

Aqui será feita a alteração das variáveis da coluna 'gear', elas passarão a ser do tipo numérico com valores variando entre 0 (automatic) e 1 (manual).

```
dados['gear'] = LabelEncoder().fit_transform(dados['gear'])
dados.tail()
```

	year_of_reference	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size	year_model	avg_price_brl
202292	2023.0	January	005538-7	ccv3mvmxnsz0dqw	VW - VolksWagen	Saveiro Robust 1.6 Total Flex 16V	Gasoline	1	1,6	2023.0	86038.0
202293	2023.0	January	005539-5	chmwf93l5hbp	VW - VolksWagen	Gol Last Edition 1.0 Flex 12V 5p	Gasoline	1	1	2023.0	95997.0
202294	2023.0	January	005539-5	cdj27stovcdqw	VW - VolksWagen	Gol Last Edition 1.0 Flex 12V 5p	Gasoline	1	1	2023.0	87828.0
202295	2023.0	January	005540-9	9w64lg6dhqp	VW - VolksWagen	Polo Track 1.0 Flex 12V 5p	Gasoline	1	1	2023.0	80845.0
202296	2023.0	January	005540-9	7hbnjm9z5dqw	VW - VolksWagen	Polo Track 1.0 Flex 12V 5p	Gasoline	1	1	2023.0	74458.0

Aqui será feita a alteração das variáveis da coluna 'fuel', elas passarão a ser do tipo numérico com valores variando entre 0 (Alcohol), 1 (Diesel) e 2 (Gasoline).

```
[16] dados['fuel'] = LabelEncoder().fit_transform(dados['fuel'])
      dados.head()
```

	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	cfzltzfwrcp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2002.0	9162.0
1	2021.0	January	004001-0	cdqwxwpw3y2p	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2001.0	8832.0
2	2021.0	January	004001-0	cbt13xwwj1xp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2000.0	8388.0
3	2021.0	January	004001-0	cb9gct6j65r0	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	0	1	1	2000.0	8453.0
4	2021.0	January	004003-7	g15wg0gbz1fx	GM - Chevrolet	Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI	2	1	1,6	2001.0	12525.0

Aqui será feita a alteração das variáveis da coluna 'brand', elas passarão a ser do tipo numérico com um numero limitado de opções. A marcas assumirão valores: 0 (Fiat), 1 (Ford), 2 (GM), 3 (Nissan), 4 (Renault) e 5 (VW).

```
[17] dados['brand'] = LabelEncoder().fit_transform(dados['brand'])
```

Aqui é feito a alteração da coluna 'engine_size', o primeiro passo é alterar o caracter "." pelo caracter ",", pois o metodo que faz a conversão de texto para valores numericos não soube lidar com a maneira como o Português lida com casas decimais, pois isso, é necessário alterar o caracter para que o Python entenda que se trata de um numero decimal. Em seguida o metodo "pandas.to_numeric()" é usado para converter os valores da coluna 'engine_size'.

```
[18] dados['engine_size'] = dados['engine_size'].str.replace('.',',')
      dados['engine_size - numeric'] = pd.to_numeric(dados['engine_size'])
      dados.head()
```

	year_of_reference	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size	year_model	avg_price_brl	engine_size - numeric
0	2021.0	January	004001-0	cfzltzfwrcp	2	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2002.0	9162.0	1.0
1	2021.0	January	004001-0	cdqwxwpw3y2p	2	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2001.0	8832.0	1.0
2	2021.0	January	004001-0	cbt13xwwj1xp	2	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2000.0	8388.0	1.0
3	2021.0	January	004001-0	cb9gct6j65r0	2	Corsa Wind 1.0 MPFI / EFI 2p	0	1	1	2000.0	8453.0	1.0
4	2021.0	January	004003-7	g15wg0gbz1fx	2	Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI	2	1	1,6	2001.0	12525.0	1.6

Aqui é criado um novo dataset utilizando somente os valores que serão utilizados no treinamento dos modelos.

```
0s dados_num = dados.drop(['month_of_reference', 'fipecode', 'authentication', 'model', 'engine_size'], axis=1)
dados_num
```

	year_of_reference	brand	fuel	gear	year_model	avg_price_brl	engine_size - numeric
0	2021.0	2	2	1	2002.0	9162.0	1.0
1	2021.0	2	2	1	2001.0	8832.0	1.0
2	2021.0	2	2	1	2000.0	8388.0	1.0
3	2021.0	2	0	1	2000.0	8453.0	1.0
4	2021.0	2	2	1	2001.0	12525.0	1.6
...
202292	2023.0	5	2	1	2023.0	86038.0	1.6
202293	2023.0	5	2	1	2023.0	95997.0	1.0
202294	2023.0	5	2	1	2023.0	87828.0	1.0
202295	2023.0	5	2	1	2023.0	80845.0	1.0
202296	2023.0	5	2	1	2023.0	74458.0	1.0

202297 rows x 7 columns

2 b)

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

Aqui eu vou definir as variáveis numéricas que serão utilizadas na análise com exceção da variável target.

```
0s [20] X = dados_num.drop('avg_price_brl', axis=1)
```

Aqui vai ser definida a variável Y apenas com a variável target.

```
0s [21] Y = dados_num['avg_price_brl']
```

Aqui é feita a divisão dos dados entre a parte que será usada para treinar o modelo e a parte que será usada para testar o modelo.

```
0s [22] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=42)
```

2 c)

c. 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

```
✓ 25s ▶ # Preencher os valores ausentes em X_train e X_test
imputer = SimpleImputer(strategy='mean')
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)

# Preencher os valores ausentes em Y_train
Y_train_imputed = imputer.fit_transform(Y_train.values.reshape(-1, 1)).ravel()

# Modelo RandomForest
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train_imputed, Y_train_imputed)

# Modelo XGBoost
xgb_model = XGBRegressor(random_state=42)
xgb_model.fit(X_train_imputed, Y_train_imputed)

# Modelo RandomForest com parâmetros ajustados
rf_model_tuned = RandomForestRegressor(n_estimators=100, max_depth=5, random_state=42)
rf_model_tuned.fit(X_train_imputed, Y_train_imputed)

# Modelo XGBoost com parâmetros ajustados
xgb_model_tuned = XGBRegressor(n_estimators=100, max_depth=3, learning_rate=0.1, random_state=42)
xgb_model_tuned.fit(X_train_imputed, Y_train_imputed)

# Avaliação dos modelos (opcional)
# (Você pode adicionar aqui a avaliação dos modelos usando X_test_imputed e Y_test)
```

```
✎ 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,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=3, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=100, n_jobs=None,
              num_parallel_tree=None, random_state=42, ...)
```

2 d)

- ✓ d. Grave os valores preditos em variáveis criadas

```
✓ 3s [24] Y_pred_rf = rf_model.predict(X_test_imputed)
      Y_pred_xgb = xgb_model.predict(X_test_imputed)
      Y_pred_rf_tuned = rf_model_tuned.predict(X_test_imputed)
      Y_pred_xgb_tuned = xgb_model_tuned.predict(X_test_imputed)
```

2 e)

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

```
✓ [25] # Análise de importância das variáveis para o modelo RandomForest
0s importances_rf = rf_model.feature_importances_
    feature_names = X.columns

    # Criar um DataFrame para visualização
    importances_df_rf = pd.DataFrame({'Feature': feature_names, 'Importance': importances_rf})
    importances_df_rf = importances_df_rf.sort_values(by='Importance', ascending=False)

    # Visualizar a importância das variáveis
    print("Importância das variáveis para o modelo RandomForest:")
    print(importances_df_rf)

    # Análise de importância das variáveis para o modelo XGBoost
    importances_xgb = xgb_model.feature_importances_

    # Criar um DataFrame para visualização
    importances_df_xgb = pd.DataFrame({'Feature': feature_names, 'Importance': importances_xgb})
    importances_df_xgb = importances_df_xgb.sort_values(by='Importance', ascending=False)

    # Visualizar a importância das variáveis
    print("Importância das variáveis para o modelo XGBoost:")
    print(importances_df_xgb)
```

```
Importância das variáveis para o modelo RandomForest:
      Feature  Importance
5  engine_size - numeric    0.481831
4      year_model    0.407889
3          gear    0.037368
2          fuel    0.033648
1          brand    0.025986
0  year_of_reference    0.013278
Importância das variáveis para o modelo XGBoost:
      Feature  Importance
5  engine_size - numeric    0.407052
4      year_model    0.275560
2          fuel    0.133859
3          gear    0.129480
1          brand    0.034835
0  year_of_reference    0.019213
```

2 f) Nos dois modelos analisados, notou-se uma maior importância para as variáveis engine_size e year_model. As demais variáveis, fuel e gear afetam de formas diferentes cada modelo. Já brand e year_of_reference são variáveis que tem uma menor influência nos dois modelos.

2 g)

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

✓
0s

```

mse = mean_squared_error(Y_test, Y_pred_rf)
mae = mean_absolute_error(Y_test, Y_pred_rf)
r2 = r2_score(Y_test, Y_pred_rf)

print("RandomForest - MSE - ", mse)
print("RandomForest - MAE -", mae)
print("RandomForest - R^2 -", r2)

mse = mean_squared_error(Y_test, Y_pred_rf_tuned)
mae = mean_absolute_error(Y_test, Y_pred_rf_tuned)
r2 = r2_score(Y_test, Y_pred_rf_tuned)

print("\nRandomForest Tuned - MSE - ", mse)
print("RandomForest Tuned - MAE -", mae)
print("RandomForest Tuned - R^2 -", r2)

mse = mean_squared_error(Y_test, Y_pred_xgb)
mae = mean_absolute_error(Y_test, Y_pred_xgb)
r2 = r2_score(Y_test, Y_pred_xgb)

print("\nXGBoost - MSE - ", mse)
print("XGBoost - MAE -", mae)
print("XGBoost - R^2 -", r2)

mse = mean_squared_error(Y_test, Y_pred_xgb_tuned)
mae = mean_absolute_error(Y_test, Y_pred_xgb_tuned)
r2 = r2_score(Y_test, Y_pred_xgb_tuned)

print("\nXGBoost Tuned - MSE - ", mse)
print("XGBoost Tuned - MAE -", mae)
print("XGBoost Tuned - R^2 -", r2)

# O melhor modelo é o RandomForest
# RandomForest - MSE - 109260296.40292338
# RandomForest - MAE - 5577.5337231235135
# RandomForest - R^2 - 0.9588152066232576
```



```
RandomForest - MSE - 109260296.40292338
RandomForest - MAE - 5577.5337231235135
RandomForest - R^2 - 0.9588152066232576

RandomForest Tuned - MSE - 424656702.10737133
RandomForest Tuned - MAE - 11730.560029104634
RandomForest Tuned - R^2 - 0.8399290583301677

XGBoost - MSE - 110781845.68670912
XGBoost - MAE - 5690.20555838328
XGBoost - R^2 - 0.9582416708108143

XGBoost Tuned - MSE - 205312790.36170554
XGBoost Tuned - MAE - 7709.001292769865
XGBoost Tuned - R^2 - 0.9226089885618961
```

2 h) O modelo que gerou o melhor resultado foi o RandomForest, utilizando os métodos de avaliação MSE (menor média de erro quadrático 109260296.40292338), MAE (menor média de erro absoluto 5577.5337231235135) e R² (maior precisão 0.9588152066232576).