

### Resolução de Problema

Você foi alocado(a) em um time da Indicium que está trabalhando atualmente junto a um cliente que o core business é compra e venda de veículos usados. Essa empresa está com dificuldades na área de revenda dos automóveis usados em seu catálogo.

Para resolver esse problema, a empresa comprou uma base de dados de um marketplace de compra e venda para entender melhor o mercado nacional, de forma a conseguir especificar o seu catálogo de forma mais competitiva e assim recuperar o mau desempenho neste setor.

Seu objetivo é analisar os dados para responder às perguntas de negócios feitas pelo cliente e criar um modelo preditivo que especifique os carros do cliente de forma que eles fiquem o mais próximos dos valores de mercado. #####

Irei analisar a base de dados do marketplace de compra e venda de carros e responder às perguntas de negócios com insights relevantes.

Além disso, desenvolverei um modelo preditivo que permita especificar os carros do cliente de forma mais competitiva, buscando se aproximar dos valores de mercado.

Aqui estão as etapas gerais que irei seguir para abordar o problema:

Análise exploratória dos dados: Primeiramente, farei uma exploração detalhada da base de dados do marketplace. Isso envolverá a compreensão dos diferentes atributos disponíveis, a identificação de quais informações são mais relevantes para a especificação e a verificação da qualidade dos dados.

Identificação de padrões e insights de negócios: Com a análise exploratória, buscarei identificar padrões, tendências e insights relevantes que possam ser úteis para a empresa cliente. Isso inclui a compreensão das preferências do mercado, fatores que influenciam os preços dos veículos usados, sazonalidade, entre outros.

Precificação competitiva: Com base nos insights obtidos, desenvolverei um modelo preditivo que leve em consideração os atributos relevantes para a especificação de veículos usados. O objetivo é criar um modelo que possa ajudar a empresa cliente a definir preços mais competitivos, alinhados com o mercado, a fim de impulsionar as vendas e melhorar o desempenho.

Avaliação do modelo: Após criar o modelo preditivo, é importante avaliá-lo adequadamente para garantir sua eficácia e precisão na especificação dos carros. Utilizarei métricas apropriadas para medir o desempenho do modelo e, se necessário, farei ajustes para otimizar seus resultados.

Implementação e suporte: Com o modelo desenvolvido e avaliado, o próximo passo é implementá-lo na empresa cliente. Além disso, estarei disponível para fornecer suporte contínuo, caso surjam dúvidas ou seja necessário ajustar o modelo com base no feedback e nas mudanças do mercado ao longo do tempo.

Vale ressaltar que, para obter resultados mais precisos e confiáveis, é fundamental trabalhar em estreita colaboração com a equipe do cliente, compartilhar informações e feedbacks constantemente. Além disso, a proteção e privacidade dos dados serão rigorosamente respeitadas ao longo de todo o processo.

Com esse plano de ação, esperamos ajudar a empresa cliente a melhorar a revenda de veículos usados e alcançar resultados mais positivos nesse setor.

```
#conexão com o google drive
from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

#importando bibliotecas
import numpy as np
import cv2
from google.colab.patches import cv2_imshow

# #caminho da imagem
img = cv2.imread('/content/drive/MyDrive/Colab Notebooks/carros.jpg')
cv2_imshow(img)
```



```
# #caminho da imagem
img1 = cv2.imread('/content/drive/MyDrive/Dicionario_dados.PNG')
cv2_imshow(img1)
```

## Dicionário de dados

**id:** Contém o identificador único dos veículos cadastrados na base de dados  
**num\_fotos:** contém a quantidade de fotos que o anúncio do veículo contém  
**marca:** Contém a marca do veículo anunciado  
**modelo:** Contém o modelo do veículo anunciado  
**versao:** Contém as descrições da versão do veículo anunciado. Sua cilindrada, quantidade de válvulas, se é flex ou não, etc.  
**ano\_de\_fabricacao:** Contém o ano de fabricação do veículo anunciado  
**ano\_modelo:** Contém o modelo do ano de fabricação do veículo anunciado  
**hodometro:** Contém o valor registrado no hodômetro do veículo anunciado  
**cambio:** Contém o tipo do câmbio do veículo anunciado  
**num\_portas:** Contém a quantidade de portas do veículo anunciado  
**tipo:** Contém o tipo do veículo anunciado. Se ele é sedã, hatch, esportivo, etc.  
**blindado:** Contém informação se o veículo anunciado é blindado ou não  
**cor:** Contém a cor do veículo anunciado  
**tipo\_vendedor:** Contém informações sobre o tipo do vendedor do veículo anunciado. Se é pessoa física (PF) ou se é pessoa jurídica (PJ)  
**cidade\_vendedor:** Contém a cidade em que vendedor do veículo anunciado reside  
**estado\_vendedor:** Contém o estado em que vendedor do veículo anunciado reside  
**anunciante:** Contém o tipo de anunciante do vendedor do veículo anunciado. Se ele é pessoa física, loja, concessionário, etc.  
**entrega\_delivery:** Contém informações se o vendedor faz ou não delivery do veículo anunciado  
**troca:** Contém informações se o veículo anunciado já foi trocado anteriormente  
**elegivel\_revisao:** Contém informações se o veículo anunciado precisa ou não de revisão  
**dono\_acelta\_troca:** Contém informações se o vendedor aceita ou não realizar uma troca com o veículo anunciado  
**veiculo\_unico\_dono:** Contém informações se o veículo anunciado é de um único dono  
**revisoes\_concessionaria:** Contém informações se o veículo anunciado teve suas revisões feitas em concessionárias  
**ipva\_pago:** Contém informações se o veículo anunciado está com o IPVA pago ou não  
**veiculo\_alienado:** Contém informações se o veículo anunciado está com o licenciamento pago ou não  
**garantia\_fabrica:** Contém informações se o veículo anunciado possui garantia de fábrica ou não  
**revisoes\_dentro\_agenda:** Contém informações se as revisões feitas do veículo anunciado foram realizadas dentro da agenda prevista  
**veiculo\_alienado:** Contém informações se o veículo anunciado está alienado ou não  
**preco (target):** Contém as informações do preço do veículo anunciado

```
...
Importação dos módulos e funções
...
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import pylab
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import homogeneity_completeness_v_measure
from sklearn.manifold import TSNE
from sklearn.preprocessing import MinMaxScaler, RobustScaler
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import KFold
```

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
import statsmodels.api as sm
import warnings
import seaborn as sns
import imblearn
from scipy import stats
from yellowbrick.cluster import KElbowVisualizer
import missingno as msno
warnings.filterwarnings("ignore")
%matplotlib inline
sns.set()
```

## 1. Análise Estatística e EDA:

### a) Estatísticas Descritivas:

Para cada variável (feature) numérica, calcule as principais estatísticas descritivas, como média, mediana, desvio padrão, mínimo, máximo e quartis. Isso ajudará a entender a distribuição e a dispersão dos dados.

```
...
Carregando os arquivos .csv como panda dataframe
...
train= pd.read_excel("/content/drive/MyDrive/train.xlsx")
train
```

```

id  num_fotos      marca     modelo      versao  ano_de_fa
n  3 007162e+38      80      NISSAN      KICKS   1.6 16V
                                             FIXSTART SI

#Analise descritiva (preliminar) automatizada
train['modelo'] = train['modelo'].astype(str)
train['modelo'] = pd.to_numeric(train['modelo'], errors='coerce')
import pandas as pd
!pip install sweetviz

import sweetviz as sv

# Caminho para o arquivo Excel
train_path = "/content/drive/MyDrive/train.xlsx"

# Carregar os dados do arquivo Excel
train = pd.read_excel(train_path)

# Converter a coluna "modelo" para uma string
train['modelo'] = train['modelo'].astype(str)

# Gerar o relatório com o Sweetviz
report = sv.analyze(train)

# Salvar o relatório em HTML
report.show_html("/content/drive/MyDrive/relatorio_sweetviz.html")

print("Relatório Sweetviz salvo em HTML com sucesso.")

```

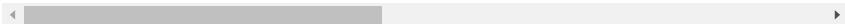
```

Requirement already satisfied: sweetviz in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: pandas!=1.0.0,!>1.0.1,!>1.0.2,>=0.25.3 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: matplotlib>=3.1.3 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: tqdm>=4.43.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: jinja2>=2.11.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: importlib-resources>=1.2.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages

```

Done! Use 'show' commands to display/save. [100%] 00:01 -> (00:00 left)

Report /content/drive/MyDrive/relatorio\_sweetviz.html was generated! NOTEBOOK/COLA  
Relatório Sweetviz salvo em HTML com sucesso.



```

!pip install pdfkit
import pandas as pd
import sweetviz as sv
import pdfkit

# Caminho para o arquivo Excel
train_path = "/content/drive/MyDrive/train.xlsx"

# Carregar os dados do arquivo Excel
train = pd.read_excel(train_path)

# Converter a coluna "modelo" para uma string
train['modelo'] = train['modelo'].astype(str)

# Gerar o relatório com o Sweetviz
report = sv.analyze(train)

# Salvar o relatório em HTML
html_path = "/content/drive/MyDrive/relatorio_sweetviz.html"
report.show_html(html_path)

print("Relatório Sweetviz gerado.")

```

Requirement already satisfied: pdfkit in /usr/local/lib/python3.10/dist-packages (0.12.0)  
 Done! Use 'show' commands to display/save.

```
train.describe()
```

	<b>id</b>	<b>num_fotos</b>	<b>ano_de_fabricacao</b>	<b>ano_modelo</b>	<b>hodometro</b>	<b>versao</b>
<b>count</b>	2.958400e+04	29407.000000	29584.000000	29584.000000	29584.000000	2
<b>mean</b>	1.705650e+38	10.323834	2016.758552	2017.808985	58430.592077	
<b>std</b>	9.814219e+37	3.487334	4.062422	2.673930	32561.769309	
<b>min</b>	1.332604e+34	8.000000	1985.000000	1997.000000	100.000000	
<b>25%</b>	8.617510e+37	8.000000	2015.000000	2016.000000	31214.000000	
<b>50%</b>	1.706531e+38	8.000000	2018.000000	2018.000000	57434.000000	
<b>75%</b>	2.554710e+38	14.000000	2019.000000	2020.000000	81953.500000	
<b>max</b>	3.402560e+38	21.000000	2022.000000	2023.000000	390065.000000	

```
test= pd.read_excel("/content/drive/MyDrive/test.xlsx")
test
```

	<b>id</b>	<b>num_fotos</b>	<b>marca</b>	<b>modelo</b>	<b>versao</b>	<b>ano_de_fabri</b>
0	1.351878e+37	8.0	NISSAN	VERSA	1.6 16V FLEXSTART V-DRIVE MANUAL	
1	2.998962e+38	18.0	FIAT	STRADA	1.4 MPI WORKING CS 8V FLEX 2P MANUAL	
2	3.161806e+38	8.0	AUDI	Q5	2.0 TFSI GASOLINA BLACK S TRONIC	
3	2.225272e+38	16.0	CHEVROLET	CRUZE	1.4 TURBO LT 16V FLEX 4P AUTOMÁTICO	
4	1.604603e+38	8.0	FORD	ECOSPORT	1.5 TI-VCT FLEX SE AUTOMÁTICO	
...	...	...	...	...	...	...
9857	2.705310e+38	8.0	TOYOTA	COROLLA	1.8 GLI UPPER 16V FLEX 4P AUTOMÁTICO	
9858	3.544423e+37	17.0	TOYOTA	HILUX	2.8 SRV 4X4 CD 16V DIESEL 4P AUTOMÁTICO	
9859	1.519276e+38	8.0	CHEVROLET	TRACKER	1.0 TURBO FLEX LT AUTOMÁTICO	
9860	1.180478e+38	9.0	PEUGEOT	2008	1.6 16V FLEX ALLURE PACK 4P AUTOMÁTICO	
9861	3.327850e+38	8.0	TOYOTA	HILUX	2.8 SRX 4X4 CD 16V DIESEL 4P AUTOMÁTICO	

9862 rows × 28 columns



```
for i, column_name in enumerate(train.columns):
  print(f"Column {i}: {column_name}")
```

```
Column 0: id
Column 1: num_fotos
Column 2: marca
```

```

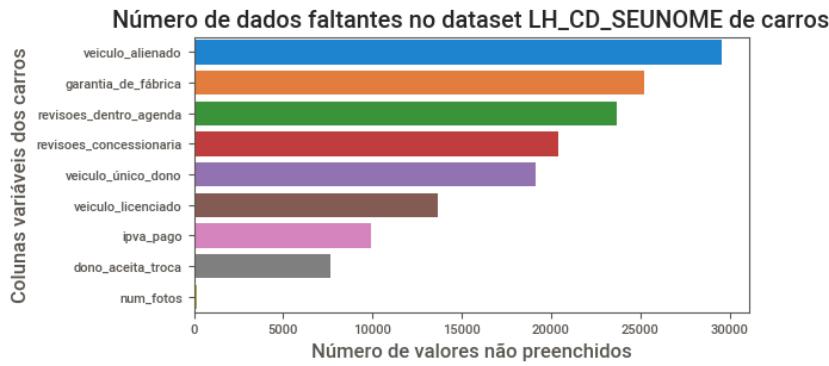
Column 3: modelo
Column 4: versao
Column 5: ano_de_fabricacao
Column 6: ano_modelo
Column 7: hodometro
Column 8: cambio
Column 9: num_portas
Column 10: tipo
Column 11: blindado
Column 12: cor
Column 13: tipo_vendedor
Column 14: cidade_vendedor
Column 15: estado_vendedor
Column 16: anunciante
Column 17: entrega_delivery
Column 18: troca
Column 19: elegivel_revisao
Column 20: dono_aceita_troca
Column 21: veiculo_único_dono
Column 22: revisoes_concessionaria
Column 23: ipva_pago
Column 24: veiculo_licenciado
Column 25: garantia_de_fábrica
Column 26: revisoes_dentro_agenda
Column 27: veiculo_alienado
Column 28: preco

```

```
X = train.isna().sum().sort_values(ascending=False)
```

```
# Filtering out columns with no missing values
X = X[X > 0]
```

```
# Creating the bar plot
plt.figure(figsize=(6, 3)) # Specify the desired figure size here (width, height)
sns.barplot(y=X.index, x=X.values, order=X.index)
plt.xlabel('Número de valores não preenchidos')
plt.ylabel('Colunas variáveis dos carros')
plt.title('Número de dados faltantes no dataset LH_CD_SEUNOME de carros')
plt.show()
```



```
train.shape
```

```
#Linhas 29584, colunas 29
```

```
(29584, 29)
```

```
for column in train.columns:
    # Usando value_counts() para contar os valores distintos e mostrar o resultado
    print(train[column].value_counts())
```

```

Name: estado_vendedor, dtype: int64
Pessoa Física          17999
Loja                   9879
Concessionária          1702
Acessórios e serviços para autos      4
Name: anuntiante, dtype: int64
False      23601
True       5983
Name: entrega_delivery, dtype: int64
False      24523
True       5061
Name: troca, dtype: int64
False      29584
Name: elegivel_revisao, dtype: int64
Aceita troca      21922
Name: dono_aceita_troca, dtype: int64
Único dono      10423
Name: veiculo_único_dono, dtype: int64
Todas as revisões feitas pela concessionária     9172
Name: revisoes_concessionaria, dtype: int64
IPVA pago      19659
Name: ipva_pago, dtype: int64
Licenciado      15906
Name: veiculo_licenciado, dtype: int64
Garantia de fábrica      4365
Name: garantia_de_fábrica, dtype: int64
Todas as revisões feitas pela agenda do carro     5910
Name: revisoes_dentro_agenda, dtype: int64
Series([], Name: veiculo_alienado, dtype: int64)
7473259008373930      1
9159064294214690      1
1515658785087030      1
2239269909210560      1
16776575600627500      1
..
343976640460006      1
28719260904618300      1
1379983440960510      1
7661572975536530      1
132508787318702800      1
Name: preco. Length: 29584. dtvne: int64

```

```
# Removendo linhas com valores vazios em todas as colunas listadas
```

```
train.dropna(subset=['num_fotos', 'marca', 'modelo', 'versao', 'ano_de_fabricacao', 'ano_modelo', 'hodometro',
                     'cambio', 'num_portas', 'tipo', 'cor', 'tipo_vendedor', 'cidade_vendedor',
                     'estado_vendedor', 'anuntiante', 'entrega_delivery', 'troca', 'elegivel_revisao'], inplace=True)
```

Qual o melhor estado cadastrado na base de dados para se vender um carro de marca popular e por quê?

Qual o melhor estado para se comprar uma picape com transmissão automática e por quê?

Qual o melhor estado para se comprar carros que ainda estejam dentro da garantia de fábrica e por quê?

```
#Qual o melhor estado cadastrado na base de dados para se vender um carro de marca popular e por quê?
```

```
media_por_estado = train.groupby('estado_vendedor', as_index=False)[['preco']].mean()
media_por_estado['preco'] = media_por_estado['preco'].astype(int)
media_por_estado['preco'] = media_por_estado['preco']/10000000000
media_por_estado
```

```
media_por_estado_ordem_decrecente = media_por_estado.sort_values(by='preco', ascending=False)
```

```
print(media_por_estado_ordem_decrecente)
```

	estado_vendedor	preco
7	Maranhão (MA)	121041.689127
0	Acre (AC)	114683.725409
22	Sergipe (SE)	114012.698831
24	Tocantins (TO)	112448.390606
15	Piauí (PI)	108627.305088
3	Bahia (BA)	104625.186698
4	Ceará (CE)	104336.181846
17	Rio Grande do Sul (RS)	104158.502667
11	Paraná (PR)	103934.028810
13	Pará (PA)	102168.831587
14	Pernambuco (PE)	100691.080335
23	São Paulo (SP)	100267.955736
16	Rio Grande do Norte (RN)	99993.993646
21	Santa Catarina (SC)	99349.763938
2	Amazonas (AM)	99258.173026
18	Rio de Janeiro (RJ)	98310.362210
10	Minas Gerais (MG)	97563.782511
6	Goiás (GO)	96250.230645
1	Alagoas (AL)	92689.970675
12	Paraíba (PB)	88737.785450
5	Espírito Santo (ES)	88669.437724
8	Mato Grosso (MT)	85842.003927

```
9  Mato Grosso do Sul (MS)  77928.295010
20          Roraima (RR)  63613.691193
19          Rondônia (RO)  33143.907028
```

```
#Qual o melhor estado para se comprar uma picape com transmissão automática e por quê?
# Filtrar apenas os registros com câmbio igual a "automático"
train_filtrado = train[train['cambio'] == 'automático']
```

```
# Agrupar por estado e calcular a média do preço
```

```
media_por_estado_ordem_decrecente = media_por_estado.sort_values(by='preco', ascending=False, inplace=False)
```

```
print(media_por_estado_ordem_decrecente)
```

	cidade_vendedor	preco
6	Almirante Tamandaré	536459.849207
8	Alvorada	490273.782949
370	Paulo Afonso	450272.363197
205	Gurupi	396559.250866
50	Barbalha	359172.417193
..	...	...
366	Pará de Minas	2832.029429
322	Monte Carmelo	2276.984821
29	Araçariguama	1326.016244
286	Luiz Alves	1061.366136
377	Perdigão	981.550351

```
[574 rows x 2 columns]
```

```
# # Filtrar apenas os registros onde a coluna "garantia_de_fábrica" é diferente de NaN
train_filtrado = train[train['garantia_de_fábrica'].notna()]
```

```
# # Supondo que você já tenha importado o pandas e o DataFrame 'train'
```

```
# # Agrupar por estado_vendedor e calcular a média do preço
```

```
media_por_estado_vendedor = train_filtrado.groupby('estado_vendedor', as_index=False)['preco'].mean()
```

```
# # Ordenar o DataFrame pelo preço em ordem decrescente
```

```
media_por_estado_vendedor = media_por_estado_vendedor.sort_values(by='preco', ascending=False)
```

```
print(media_por_estado_vendedor)
```

	estado_vendedor	preco
20	Tocantins (TO)	2.430022e+16
0	Acre (AC)	1.504169e+16
18	Sergipe (SE)	1.259448e+16
12	Pará (PA)	1.204898e+16
8	Mato Grosso do Sul (MS)	1.142514e+16
3	Bahia (BA)	1.126614e+16
17	Santa Catarina (SC)	1.113455e+16
10	Paraná (PR)	1.112516e+16
15	Rio Grande do Sul (RS)	1.097969e+16
19	São Paulo (SP)	1.026779e+16
6	Goiás (GO)	1.014558e+16
7	Mato Grosso (MT)	1.002860e+16
16	Rio de Janeiro (RJ)	1.000878e+16
11	Paraíba (PB)	9.576275e+15
9	Minas Gerais (MG)	9.165725e+15
14	Rio Grande do Norte (RN)	8.834099e+15
1	Alagoas (AL)	8.336154e+15
13	Pernambuco (PE)	7.695018e+15
2	Amazonas (AM)	7.548556e+15
5	Espírito Santo (ES)	6.796960e+15
4	Ceará (CE)	1.239399e+15

```
# Obter todas as alternativas únicas na coluna "modelo"
```

```
alternativas_modelo = train['modelo'].unique()
```

```
print(alternativas_modelo)
```

```
'X6' 'A 200' 'T4' 'V60' 'COROLLA' 'RX350 F-SPORT' 'ASX' 'LANCER' 'TIGUAN'
'SAVEIRO' '208' 'GOLF' 'KANGOO' 'C 180' 'IX35' 'CRUZE' 'RENEGADE'
'STRADA' 'M3' 'A5' 'FIORINO' '320i' 'CRV' 'TRACKER' 'TORO' 'RAV4'
'COOPER' 'RANGE ROVER EVOQUE' 'VIRTUS' 'M5' 'JETTA' '2008'
'RANGE ROVER SPORT' 'SPIN' 'CLS 350' 'GLA 200' 'GOL' 'VOYAGE'
'COUNTRYMAN' 'TIGGO 2' 'DISCOVERY SPORT' 'HILUX' 'RANGER' 'PALIO' 'FOX'
'A4' 'MONTANA' 'YARIS' 'RANGE ROVER VELAR' 'MACAN' 'T-CROSS' 'S60'
'CLASSIC' 'DUSTER' 'TERRITORY' 'TUCSON' 'CADENZA' 'PRISMA' 'JOY'
'CAPTIVA' 'CERATO' 'CRONOS' 'KA' 'SPRINTER' 'FRONTIER' 'C 180 K' 'HB20'
'NEW BEETLE' 'SANDERO' 'A6' 'DOBLÔ' 'CRETA' 'S10' '328i' 'MASTER'
'SPORTAGE' 'XC60' 'VERSA' 'L200 TRITON' 'CLIO' 'DISCOVERY' 'FUSCA'
'C 250' 'CT200H' 'GLK 220' 'ELANTRA' 'E 250' 'TOUAREG' 'CAPTUR' 'CITY'
'DUSTER OROCH' 'TAOS' 'Q5' 'Q7' 'ECLIPSE' 'WRANGLER' 'ECLIPSE CROSS'
```

```

PASSAI PRIUS FUSION AI AL40 COROLLA CROSS 481 MUSTANG
'COUPÉ' 'MALIBU' 'ETIOS' 'FLUENCE' 'Camaro' 'XC90' 'LINEA' 'PANAMERA'
'LOGAN' 'IDEA' 'L200' 'A 190' 'GLA 250' 'ARRIZO 6' 'SLK 200' 'XE' 'PUNTO'
'NX 300h' '530i' 'DUCATO' 'AZERA' 'CLK 320' '120i' 'C 200' 'CLS 400' 'XF'
'RS5' 'S 560' 'ACCORD' 'SLK 250' '718' 'S-CROSS' 'FREEMONT' 'HR'
'PICANTO' 'TIGGO 5x' 'C 230 K' 'CLA 250' 'NX 300' 'FOCUS' '3008'
'CHEROKEE' 'C 63 AMG' 'POLO SEDAN' 'DISCOVERY 4' 'F-PACE' 'C 450'
'FREELANDER 2' 'C4 GRAND PICASSO' 'EDGE' 'GHIBLI' '911' 'X2'
'PASSAT VARIANT' 'ML 350' 'I30' '528i' 'AGILE' 'MOBI' 'SPACEFOX' 'MARCH'
'LIVINA' 'V40' 'X7' 'DISCOVERY 3' '207' 'B 200' 'BRAVO' 'X4' 'C4 LOUNGE'
'X3' '5008' 'RANGE ROVER VOGUE' 'OUTLANDER SPORT' 'S 55 AMG' 'NIVUS'
'CARAVAN' 'C3 PICASSO' 'CLC 200 K' 'SENTRA' 'MOHAVE' 'JOURNEY' 'S90'
'CARNIVAL' 'F-1000' '550i' 'ES 300h' 'A 35 AMG' '130i' 'GRAND LIVINA'
'300 C' 'E-PACE' '147' 'E 63 AMG' 'NX 200T' 'PAJERO SPORT' 'FIESTA'
'Kwid' 'MERIVA' 'C 350' '745Le' 'GLC 43 AMG' 'PAJERO' 'OMEGA'
'BRONCO SPORT' 'VELOSTER' 'ARRIZO 5' 'CROSSFOX' 'CLA 200' '330e'
'C 43 AMG' 'TT RS' 'SQ5' 'M6' 'C 300' 'GLK 300' 'VECTRA' 'DSS'
'AMG GT 63' 'CROSS UP' 'ZOE' 'M 340i' 'TOWN & COUNTRY' 'F-250' '330i'
'JUMPY' 'C4' '428i' 'GRAND VITARA' 'GLB 200' '125i' 'i3' 'SPACE CROSS'
'SIENA' 'ONE' 'RS Q3' 'ASTRA' 'TTS' 'COBALT' 'SANTANA' 'VITARA' 'TAYCAN'
'E-TRON' 'KOMBI' 'FORTWO' 'BLAZER' '225i' 'M 135i' 'GLE 400d' 'A 250'
'HB20X' 'TIGGO 7' 'BONGO' 'C 280' 'M2' 'ML 63 AMG' 'COMMANDER' '408'
'220i' 'SX4' 'GLC 63 AMG' 'GLC 300' 'M 235i' 'MÉGANE' 'ROADSTER'
'DEFENDER' '430i' 'XV' '307' '323ti' 'TRIBECA' 'E 350' 'A7' 'SOUL'
'IMPREZA' 'GRAND BLAZER' 'ETIOS CROSS' '335i' 'ZAFIRA' 'i8' 'CLA 180'
'LEGACY' 'DAKOTA' 'X60' 'A 45 AMG' 'C4 PICASSO' 'I-PACE' '308'
'PAJERO DAKAR' 'GL 63 AMG' 'UX 250h' 'Q8' 'E 430' 'CELTA'
'PIAPE CAÇAMBA LONGA' 'AMG GT' 'F-TYPE' 'S 500' 'RS6' 'AIRCROSS'
'XSARA PICASSO' 'VERACRUZ' 'X80' '420i' 'GLA 45 AMG' 'CORSA' 'GLE 63 AMG'
'E 300' '325i' 'TIGGO 8' 'RIO' 'EXPERT' 'BOXER' 'TIGGO 3X' 'TX4' 'EQUUS'
'JIMNY' 'B 180' '1500' 'GLS 350' 'M 240i' 'ESCORT' 'RS4' 'DAILY'
'FORESTER' '118i' 'S4' 'SLC 300' 'ES' 'QUATTROPORTE' 'CAYMAN'
'CLA 45 AMG' 'T40' 'SILVERADO' 'L200 OUTDOOR' '750i' 'X-TRAIL' 'BOXSTER'
'500' 'XTERRA' 'GRAND SANTA FÉ' 'TIIDA' 'PACEMAN' 'RS3' 'JUMPER' 'CAMRY'
'ML 320' 'F360' 'E 320' '407' 'GLE 43 AMG' 'GLE 400' 'SONIC'
'L200 SAVANA' 'BORA' 'S80' 'GLE 350' '545i' '530' 'I30 CW' 'SONATA'
'PT CRUISER' 'SL 63 AMG' 'CLS 450' 'E 55 AMG' '640i' 'SYMBOL' 'DURANGO'
'M-11' 'V70' 'FREELANDER' 'S 500 L' '156' 'T60' 'DS4' 'D20' 'CLS 53 AMG'
'530e' 'CARENS' 'Z3' '316i' 'RAM' '300 M' 'ES 350' 'STILO' 'RS7' 'S6'
'XKR' 'E 500' 'COURIER' 'RX 450h' 'SLK 55 AMG' 'C30' 'C 240' '116i'
'AIR TREK' 'M 850i' 'G 63 AMG' 'M4' 'GLC 220d' 'S40' 'RCZ' 'PASEO'
'SLK 230' ]

```

```

train.shape
#Linhas 29584 passou a ter 29407, colunas 29

(29407, 29)

```

```

!pip install tabulate
#Variáveis nominais
from tabulate import tabulate

```

```

# Supondo que 'train' é o seu DataFrame

# Obtendo a descrição das variáveis nominais
describe_nominais = train.describe(include='O')

# Exibindo o resultado em uma tabela
print(tabulate(describe_nominais, headers='keys', tablefmt='pretty'))

```

```

Requirement already satisfied: tabulate in /usr/local/lib/python3.10/dist-packages (0.9.0)
+-----+-----+-----+-----+-----+-----+-----+-----+
|     | marca | modelo |           versao          |   cambio | tipo | blindado | cor | tipo_vendedor |
+-----+-----+-----+-----+-----+-----+-----+-----+
| count | 29407 | 29407 |                 29407 | 29407 | 29407 | 29407 | 29407 | 29407
| unique |    40 |    457 |                 1916 |      7 |      7 |      2 |      7 |      2
| top   | VOLKSWAGEN | COMPASS | 1.6 16V FLEX ALLURE PACK 4P AUTOMÁTICO | Automática | Sedã | N | Branco | PF
| freq  | 4582 | 1478 |                 1271 | 22428 | 16307 | 29159 | 20848 | 17888
+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

# Calculando as estatísticas para a variável "hodometro"
estatisticas_hodometro = train['hodometro'].describe()

```

```

# Criando um DataFrame para armazenar as estatísticas
tabela_hodometro = pd.DataFrame(estatisticas_hodometro)

```

```

# Exibindo o DataFrame em formato de tabela
print(tabulate(tabela_hodometro, headers='keys', tablefmt='pretty'))

```

```

+-----+-----+
|     | hodometro |
+-----+-----+
| count | 29407.0 |
| mean  | 58578.24827422042 |
| std   | 32581.28101776799 |

```

min	100.0
25%	31316.5
50%	57707.0
75%	82095.5
max	390065.0

```
# Calculando as estatísticas para a variável "preco"
estatisticas_preco = train['preco'].describe()

# Criando um DataFrame para armazenar as estatísticas
tabela_preco = pd.DataFrame(estatisticas_preco)

# Exibindo o DataFrame em formato de tabela
print(tabulate(tabela_preco, headers='keys', tablefmt='pretty'))
```

	preco
count	29407.0
mean	1.002983822787313e+16
std	9564310495134626.0
min	20062921479.0
25%	3202475009355035.0
50%	8241621892869420.0
75%	1.342557039370815e+16
max	6.54991187114168e+16

### b) Gráficos:

Plote gráficos adequados para cada variável numérica, como histogramas, boxplots ou densidade de distribuição. Esses gráficos fornecerão insights visuais sobre a distribuição dos dados, a presença de outliers e padrões.

```
# Contagem das ocorrências de cada marca
contagem_marcas = train['marca'].value_counts()

# Obtendo as marcas únicas em ordem de contagem decrescente
marcas_unicas = contagem_marcas.index

# Definindo cores diferentes para as barras com base no número de marcas
cores = plt.cm.tab20c.colors[:len(marcas_unicas)]

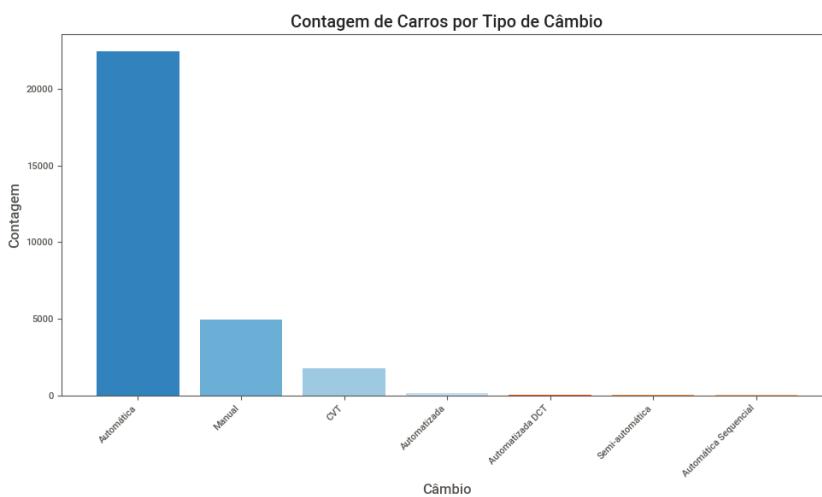
# Plotando o gráfico de barras
plt.figure(figsize=(10, 6)) # Define o tamanho da figura do gráfico (largura, altura)
plt.bar(marcas_unicas, contagem_marcas, color=cores)
plt.xlabel('Marca')
plt.ylabel('Contagem')
plt.title('Contagem de Carros por Marca')
plt.xticks(rotation=45, ha='right') # Rotaciona os rótulos do eixo x para melhor visualização
plt.tight_layout() # Ajusta o layout para evitar sobreposição de rótulos
plt.show()
```

```
Contagem de Carros por Marca
4000
# Contagem das ocorrências de cada tipo de câmbio
contagem_cambio = train['cambio'].value_counts()

# Obtendo os tipos de câmbio únicos em ordem de contagem decrescente
cambio_unicos = contagem_cambio.index

# Definindo cores diferentes para as barras com base no número de tipos de câmbio
cores_cambio = plt.cm.tab20c.colors[:len(cambio_unicos)] 

# Plotando o gráfico de barras para os tipos de câmbio
plt.figure(figsize=(10, 6)) # Define o tamanho da figura do gráfico (largura, altura)
plt.bar(cambio_unicos, contagem_cambio, color=cores_cambio)
plt.xlabel('Câmbio')
plt.ylabel('Contagem')
plt.title('Contagem de Carros por Tipo de Câmbio')
plt.xticks(rotation=45, ha='right') # Rotaciona os rótulos do eixo x para melhor visualização
plt.tight_layout() # Ajusta o layout para evitar sobreposição de rótulos
plt.show()
```

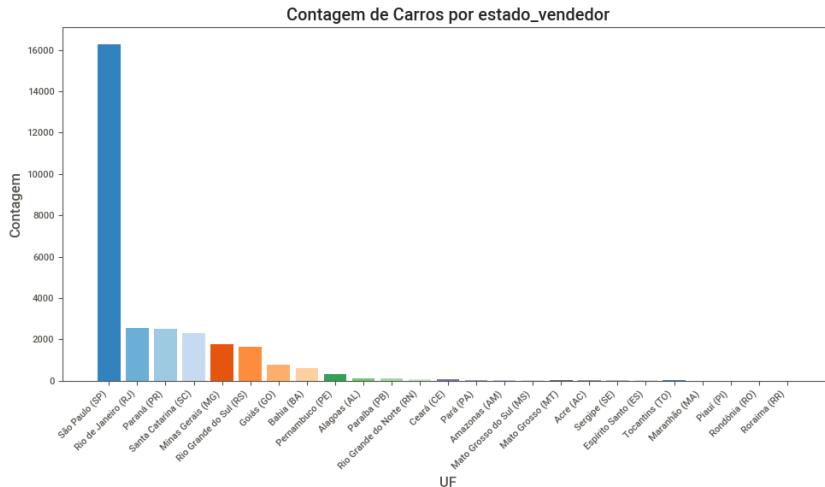


```
# Contagem das ocorrências de cada marca
contagem_UF = train['estado_vendedor'].value_counts()

# Obtendo as marcas únicas em ordem de contagem decrescente
UF_unicas = contagem_UF.index

# Definindo cores diferentes para as barras com base no número de marcas
cores = plt.cm.tab20c.colors[:len(UF_unicas)] 

# Plotando o gráfico de barras
plt.figure(figsize=(10, 6)) # Define o tamanho da figura do gráfico (largura, altura)
plt.bar(UF_unicas, contagem_UF, color=cores)
plt.xlabel('UF')
plt.ylabel('Contagem')
plt.title('Contagem de Carros por estado_vendedor')
plt.xticks(rotation=45, ha='right') # Rotaciona os rótulos do eixo x para melhor visualização
plt.tight_layout() # Ajusta o layout para evitar sobreposição de rótulos
plt.show()
```



```
from scipy.stats import gaussian_kde

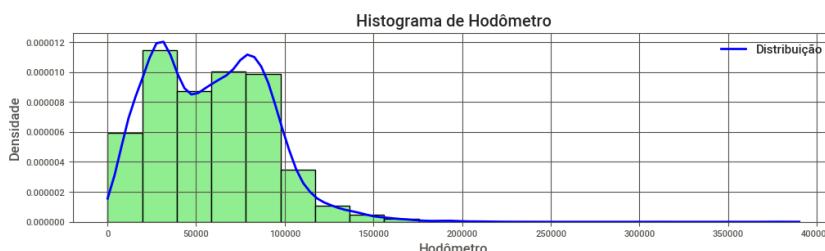
# Plotando o histograma da variável "hodometro"
plt.figure(figsize=(12, 3)) # Define o tamanho da figura do gráfico (largura, altura)
plt.hist(train['hodometro'], bins=20, color='lightgreen', edgecolor='black', density=True)
plt.xlabel('Hodômetro')
plt.ylabel('Densidade')
plt.title('Histograma de Hodômetro')

# Estimando a densidade da distribuição
densidade = gaussian_kde(train['hodometro'])
hodometro_min = train['hodometro'].min()
hodometro_max = train['hodometro'].max()
hodometro_range = np.linspace(hodometro_min, hodometro_max, 100) # Gera valores para a linha da distribuição
plt.plot(hodometro_range, densidade(hodometro_range), color='blue', linewidth=2, label='Distribuição')

plt.legend() # Mostra a legenda com o rótulo da distribuição

plt.grid(True) # Adiciona grades no gráfico

plt.show()
```



```
# Plotando o histograma da variável "preço"
plt.figure(figsize=(6, 3)) # Define o tamanho da figura do gráfico (largura, altura)
plt.hist(train['preço'], bins=20, color='skyblue', edgecolor='black', density=True)
plt.xlabel('Preço')
plt.ylabel('Densidade')
plt.title('Histograma de Preço')

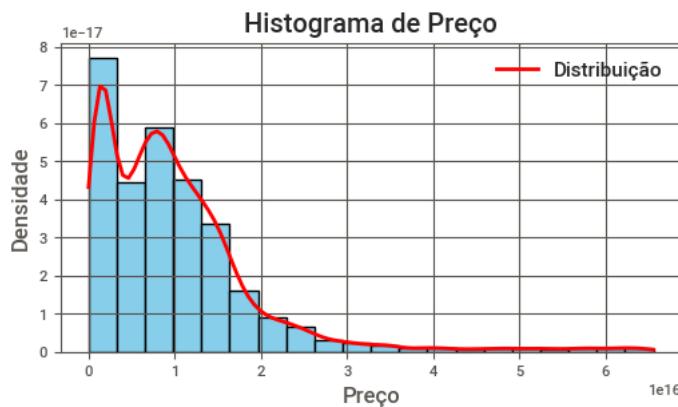
# Estimando a densidade da distribuição
densidade = gaussian_kde(train['preço'])
preço_min = train['preço'].min()
preço_max = train['preço'].max()
preço_range = np.linspace(preço_min, preço_max, 100) # Gera valores para a linha da distribuição
plt.plot(preço_range, densidade(preço_range), color='red', linewidth=2, label='Distribuição')

plt.legend() # Mostra a legenda com o rótulo da distribuição
```

```

plt.grid(True) # Adiciona grades no gráfico
plt.show()

```



c) Correlações:

Calcule a matriz de correlação entre as variáveis numéricas. Isso permitirá identificar relações lineares entre as features e a variável alvo "preco".

```

# Selecionando apenas as variáveis numéricas
variaveis_numericas = train.select_dtypes(include='number')

# Calculando a matriz de correlação
matriz_correlacao = variaveis_numericas.corr()

# Exibindo a matriz de correlação
print(matriz_correlacao)

          id  num_fotos  ano_de_fabricacao  ano_modelo \
id      1.000000   -0.003772    -0.006258   -0.000994
num_fotos     -0.003772   1.000000     0.029702    0.032821
ano_de_fabricacao  -0.006258    0.029702   1.000000    0.859698
ano_modelo     -0.000994    0.032821     0.859698   1.000000
hodometro      -0.002707    0.027550    -0.727464   -0.789745
num_portas     -0.004590    0.011245     0.080241    0.073631
veiculo_alienado  NaN        NaN        NaN        NaN
preco          0.006298   -0.005224     0.014482    0.014694

          hodometro  num_portas  veiculo_alienado  preco
id          -0.002707   -0.004590        NaN  0.006298
num_fotos      0.027550    0.011245        NaN -0.005224
ano_de_fabricacao  -0.727464    0.080241        NaN  0.014482
ano_modelo      -0.789745    0.073631        NaN  0.014694
hodometro        1.000000   -0.054970        NaN -0.027905
num_portas      -0.054970    1.000000        NaN -0.001569
veiculo_alienado  NaN        NaN        NaN        NaN
preco          -0.027905   -0.001569        NaN  1.000000

```

```

# Selecionando apenas a variável numérica 'hodometro'
preco= train['preco']

```

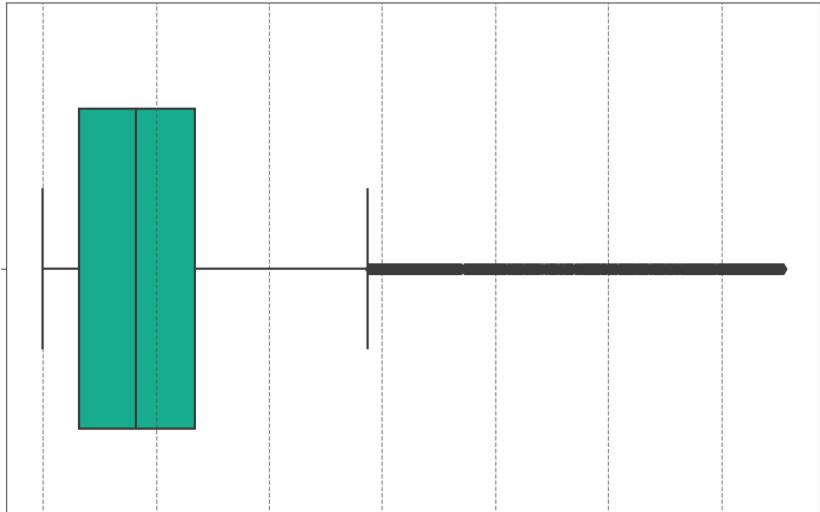
```

# Definindo cores personalizadas para o boxplot
cor_boxplot_preco = '#00c49D'

# Criando o boxplot estilizado do hodômetro
plt.figure(figsize=(8, 6)) # Define o tamanho da figura do gráfico (largura, altura)
sns.boxplot(x=preco, color=cor_boxplot_preco, orient='h', width=0.6)
plt.xlabel('Preço')
plt.title('Boxplot do Preço')
plt.grid(axis='x', linestyle='--', alpha=0.7) # Adiciona linhas de grade no eixo x
plt.xticks(fontsize=12) # Ajusta o tamanho dos rótulos do eixo x
plt.tight_layout() # Ajusta o layout para evitar sobreposição de elementos
plt.show()

```

Boxplot do Preço

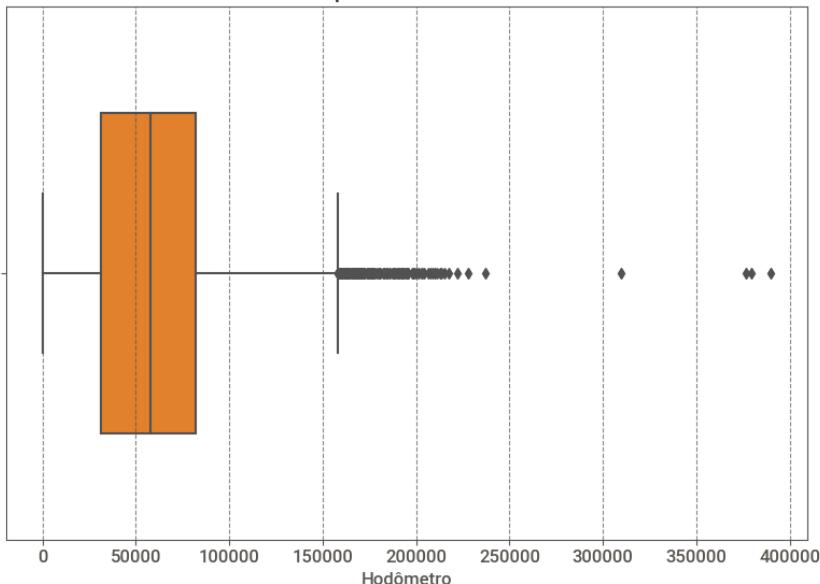


```
# Selecionando apenas a variável numérica 'hodometro'
hodometro = train['hodometro']

# Definindo cores personalizadas para o boxplot
cor_boxplot_hodo = '#ff7f0e'

# Criando o boxplot estilizado do hodômetro
plt.figure(figsize=(8, 6)) # Define o tamanho da figura do gráfico (largura, altura)
sns.boxplot(x=hodometro, color=cor_boxplot_hodo, orient='h', width=0.6)
plt.xlabel('Hodômetro')
plt.title('Boxplot do Hodômetro')
plt.grid(axis='x', linestyle='--', alpha=0.7) # Adiciona linhas de grade no eixo x
plt.xticks(fontsize=12) # Ajusta o tamanho dos rótulos do eixo x
plt.tight_layout() # Ajusta o layout para evitar sobreposição de elementos
plt.show()
```

Boxplot do Hodômetro



#### Hipóteses:

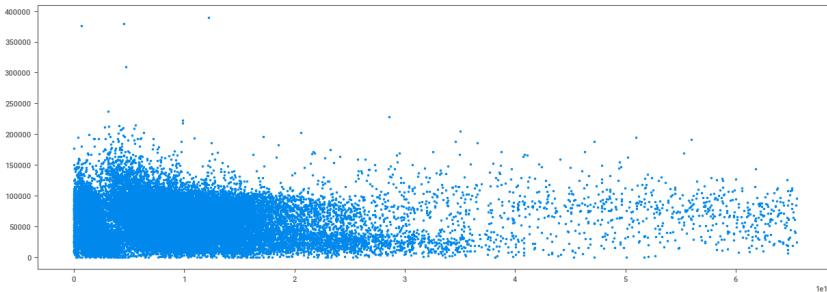
Hipótese 1: As variáveis preço pode estar relacionada com a quilometragem.

Hipótese 2: A variável câmbio automático pode estar relacionada com quilometragem e preço de venda.

Hipótese 2: Carros mais抗igos podem estar relacionados com o preço de venda.

```
#dispersao preço e quilometragem
plt.figure(figsize=(15,5))
plt.plot(train.preco,train.hodometro, '.')
```

```
[<matplotlib.lines.Line2D at 0x78b60819a5c0>]
```



```
# Supondo que 'train' é o seu conjunto de dados
plt.figure(figsize=(15, 5))

# Plotando a dispersão dos preços de carros com transmissão manual em relação à quilometragem
# Usando pontos vermelhos ('.') para representar os dados
plt.plot(train.loc[train.cambio == 'Manual', 'hodometro'], train.loc[train.cambio == 'Manual', 'preco'], '.', color='red')

# Adicionando etiqueta para o eixo x
plt.xlabel('Quilometragem (hodometro)')

# Adicionando etiqueta para o eixo y
plt.ylabel('Preço (preco)')

# Adicionando título ao gráfico
plt.title('Dispersão dos Preços de Carros Manuais\nQuilometragem vs. Preço')

# Habilitando as grades para melhor visualização dos dados
plt.grid(True)

# Exibindo o gráfico na tela
plt.show()
```



```
# Supondo que 'train' é o seu DataFrame
plt.figure(figsize=(12, 6))

# Plotando o gráfico de dispersão
plt.scatter(train['ano_de_fabricacao'], train['preco'], c=train['hodometro'], cmap='viridis', alpha=0.6)

# Adicionando etiqueta para o eixo x
```

```

plt.xlabel('Ano de Fabricação')

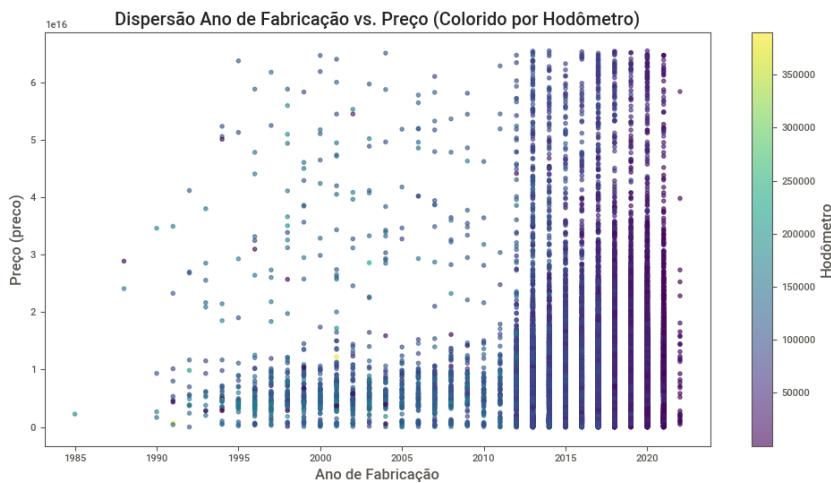
# Adicionando etiqueta para o eixo y
plt.ylabel('Preço (preco)')

# Adicionando título ao gráfico
plt.title('Dispersão Ano de Fabricação vs. Preço (Colorido por Hodômetro)')

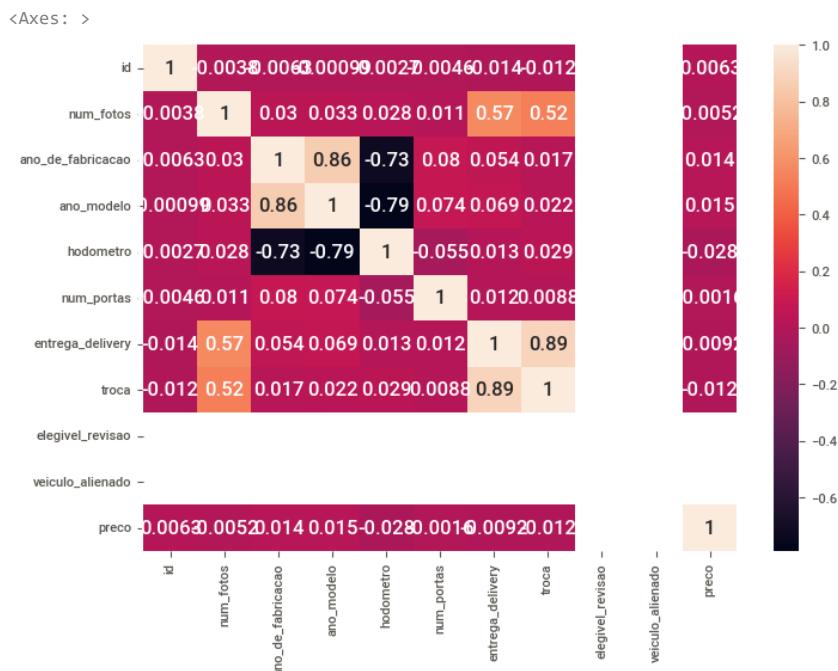
# Adicionando uma barra de cores para representar o hodômetro (quanto mais escuro, maior a quilometragem)
cbar = plt.colorbar()
cbar.set_label('Hodômetro')

# Exibindo o gráfico na tela
plt.show()

```



```
sns.heatmap(train.corr(), annot = True )
```



Respostas às Perguntas de Negócio:

a) Qual o melhor estado cadastrado na base de dados para se vender um carro de marca popular e por quê?

Resposta: Analise as estatísticas de preços e a demanda por carros de marca popular em cada estado. O melhor estado será aquele em que os carros desse tipo têm preços mais altos e/ou são vendidos mais rapidamente.

b) Qual o melhor estado para se comprar uma picape com transmissão automática e por quê?

Resposta: Analise as estatísticas de preços e a disponibilidade de picapes com transmissão automática em cada estado. O melhor estado será aquele em que esses veículos têm preços mais baixos e/ou são mais facilmente encontrados no mercado. c) Qual o melhor estado para se comprar carros que ainda estejam dentro da garantia de fábrica e por quê?

Resposta: Analise as estatísticas de preços e a oferta de carros com garantia de fábrica em cada estado. O melhor estado será aquele em que esses carros têm preços mais baixos e/ou são mais abundantes no mercado.

Mapa

## ▼ Tratando a variável

```
# Renomear coluna para 'NM_ESTADO' em letras maiúsculas
train.rename(columns={'estado_vendedor': 'NM_ESTADO'}, inplace=True)
# Remover as quatro últimas letras da coluna 'NM_ESTADO'
train['NM_ESTADO'] = train['NM_ESTADO'].str.slice(stop=-4).str.upper()
train['NM_ESTADO']

0           SAO PAULO
1        MINAS GERAIS
2           SAO PAULO
3           SAO PAULO
4      RIO DE JANEIRO
...
29579          GOIAS
29580         PARANA
29581          BAHIA
29582          SAO PAULO
29583          SAO PAULO
Name: NM_ESTADO, Length: 29584, dtype: object

#ESTADO PREÇO
media_por_estado = train.groupby('NM_ESTADO')['preco'].mean()

media_por_estado = train.groupby('NM_ESTADO', as_index=False)['preco'].mean()
media_por_estado['preco'] = media_por_estado['preco'].astype(int)
media_por_estado['preco'] = media_por_estado['preco']/100000000000
media_por_estado
```

	NM_ESTADO	preco
0	ACRE	114683.725409
1	ALAGOAS	92819.191291
2	AMAZONAS	99258.173026
3	BAHIA	104465.679742
4	CEARA	104336.181846

Previsão do Preço:

Para prever o preço dos carros, você pode usar técnicas de regressão, pois estamos resolvendo um problema de previsão numérica contínua. Algumas variáveis importantes a serem consideradas para a previsão do preço podem incluir: ano de fabricação, quilometragem, marca, modelo, tipo de veículo, estado de conservação, entre outras.

Pode-se experimentar diferentes modelos de regressão, como regressão linear, regressão por floresta aleatória, regressão de gradient boosting, ou modelos mais avançados, como redes neurais.

Avaliado o desempenho do modelo usando métricas de regressão, como o erro médio absoluto (MAE) ou o erro quadrático médio (MSE). Escolha a métrica mais adequada com base na escala dos dados e no objetivo de negócios.

Finalmente, gerar o arquivo "predição" com as previsões de preço para o conjunto de dados e compartilhar os resultados com o cliente.

Utilizando as variáveis (features), faça um relatório com uma análise das principais estatísticas da base de dados. Descreva graficamente essas variáveis (features), apresentando as suas principais estatísticas descritivas. Comente o porquê da escolha destas estatísticas e o que elas nos informam.

Faça uma EDA. Nesta EDA, crie e responda 3 hipóteses de negócio. Além disso, responda também às seguintes perguntas de negócio:

Qual o melhor estado cadastrado na base de dados para se vender um carro de marca popular e por quê?

Qual o melhor estado para se comprar uma picape com transmissão automática e por quê?

Qual o melhor estado para se comprar carros que ainda estejam dentro da garantia de fábrica e por quê?

Explique como você faria a previsão do preço a partir dos dados. Quais variáveis e/ou suas transformações você utilizou e por quê? Qual tipo de problema estamos resolvendo (regressão, classificação)?

Qual modelo melhor se aproxima dos dados e quais seus prós e contras? Qual medida de performance do modelo foi escolhida e por quê?

Envie o resultado final do modelo em uma planilha com apenas duas colunas (id, preco).

A entrega deve ser feita através de um repositório de código público que contenha:

README explicando como instalar e executar o projeto Arquivo de requisitos com todos os pacotes utilizados e suas versões Relatórios das análises estatísticas e EDA em PDF, Jupyter Notebook ou semelhante conforme passo 1 e 2. Códigos de modelagem utilizados no passo 3.

Arquivo final com o nome predicted.csv conforme passo 4 acima.

Todos os códigos produzidos devem seguir as boas práticas de codificação.

```

import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn import metrics

# Read the data
df_train = pd.read_excel("/content/drive/MyDrive/train.xlsx")
df_test = pd.read_excel("/content/drive/MyDrive/test.xlsx")

# Prepare the data
x = df_train[['ano_de_fabricacao', 'ano_modelo', 'hodometro', 'num_portas']]
y = df_train['preco']
z = df_test[['ano_de_fabricacao', 'ano_modelo', 'hodometro', 'num_portas']]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=0)

# Optionally, scale the data using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Create and train the Linear Regression model
linear_regression = LinearRegression()
linear_regression.fit(X_train_scaled, y_train)

# Make predictions on the test set
y_pred = linear_regression.predict(X_test_scaled)

# Evaluate the model

```

```
mae = metrics.mean_absolute_error(y_test, y_pred)
mse = metrics.mean_squared_error(y_test, y_pred)
r2 = metrics.r2_score(y_test, y_pred)

print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("R-squared:", r2)

#comenta e roda o print(logistic_regression.coef_) como primeiro step, depois descomenta, analisa e roda: y_pred = logistic_regression.pr
y_pred = logistic_regression.predict(z)
```

```
Mean Absolute Error: 6415761724982024.0
Mean Squared Error: 8.762770864467052e+31
R-squared: -0.00047395460785271126
```

```
print(logistic_regression.coef_)
```

```
[[ -2.64870632e-04 -2.65046396e-04  9.28761298e-05 -4.83217841e-07]
 [ -5.54759436e-04 -5.55773872e-04  1.09659390e-04 -1.04358867e-06]
 [ -2.96809487e-04 -2.97604166e-04  9.50699292e-05 -5.43187796e-07]
 ...
 [ -4.10758107e-04 -4.10457539e-04  1.01261916e-04 -7.63023154e-07]
 [  5.34325202e-04  5.34662658e-04  1.30306720e-05  1.07498758e-06]
 [ -5.78088331e-04 -5.79124544e-04  1.10886253e-04 -1.09018019e-06]]
```

---

```
✓ 21s conclusão: 17:02
```

