In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import pylab
import sys
import warnings
import statistics
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
```

In [62]:
```
```

In [2]:
```
# Comandos para suprimir os warnings
if not sys.warnoptions:
    warnings.simplefilter("ignore")
```

In [3]:
```
# Leando o arquivo

df1 = pd.read_csv('cars_train.csv', encoding='UTF-16', sep='\t')
```

In [4]:
```
df1.head(3)
```

Out[4]:

	id	num_fotos	marca	modelo	versao
0	300716223898539419613863097469899222392	8.0	NISSAN	KICKS	1.6 16V FLEXSTART SL 4P XTRONIC
1	279639842134129588306469566150288644214	8.0	JEEP	COMPASS	2.0 16V FLEX LIMITED AUTOMÁTICO
2	564144608106210489002956782365381719 81	16.0	KIA	SORENTO	2.4 16V GASOLINA EX 7L AWD AUTOMÁTICO

3 rows × 29 columns

In [5]:
```
# Explorando o arquivo parte 1
#type(df1)
```

In [6]:
```
#df1.info()
```

In [7]:
```
# Altarando o tipo de dados
df1['id'] = df1['id'].astype('category')
# df1['num_fotos'] = df1['num_fotos'].astype()
df1['marca'] = df1['marca'].astype('category')
df1['modelo'] = df1['modelo'].astype('category')
df1['versao'] = df1['versao'].astype('category')
df1['cambio'] = df1['cambio'].astype('category')
df1['tipo'] = df1['tipo'].astype('category')
df1['blindado'] = df1['blindado'].astype('category')
df1['tipo_vendedor'] = df1['tipo_vendedor'].astype('category')
df1['cidade_vendedor'] = df1['cidade_vendedor'].astype('category')
df1['estado_vendedor'] = df1['estado_vendedor'].astype('category')
df1['anunciante'] = df1['anunciante'].astype('category')
df1['dono_aceita_troca'] = df1['dono_aceita_troca'].astype('category')
df1['veiculo_licenciado'] = df1['veiculo_licenciado'].astype('category')
df1['garantia_de_fábrica'] = df1['garantia_de_fábrica'].astype('category')
df1['revisoes_dentro_agenda'] = df1['revisoes_dentro_agenda'].astype('categ
df1['cor'] = df1['cor'].astype('category')
df1['veiculo_único_dono'] = df1['veiculo_único_dono'].astype('category')
df1['revisoes_concessionaria'] = df1['revisoes_concessionaria'].astype('cat
df1['ipva_pago'] = df1['ipva_pago'].astype('category')

df1['ano_de_fabricacao'] = df1['ano_de_fabricacao'].astype(np.int16)
df1['num_portas'] = df1['num_portas'].astype(np.int16)
df1['ano_modelo'] = df1['ano_modelo'].astype(np.int16)
df1['preco'] = df1['preco'].astype(np.int32)
df1['entrega_delivery'] = df1['entrega_delivery'].astype(np.int32)
df1['troca'] = df1['troca'].astype(np.int32)
df1['elegivel_revisao'] = df1['elegivel_revisao'].astype(np.int32)
```

In [8]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29584 entries, 0 to 29583
Data columns (total 29 columns):
 #   Column                   Non-Null Count  Dtype   
---  ------                   --------------  -----   
 0   id                       29584 non-null  category
 1   num_fotos                29407 non-null  float64 
 2   marca                    29584 non-null  category
 3   modelo                   29584 non-null  category
 4   versao                   29584 non-null  category
 5   ano_de_fabricacao        29584 non-null  int16   
 6   ano_modelo               29584 non-null  int16   
 7   hodometro                29584 non-null  float64 
 8   cambio                   29584 non-null  category
 9   num_portas               29584 non-null  int16   
 10  tipo                     29584 non-null  category
 11  blindado                 29584 non-null  category
 12  cor                      29584 non-null  category
 13  tipo_vendedor            29584 non-null  category
 14  cidade_vendedor          29584 non-null  category
 15  estado_vendedor          29584 non-null  category
 16  anunciante               29584 non-null  category
 17  entrega_delivery         29584 non-null  int32   
 18  troca                    29584 non-null  int32   
 19  elegivel_revisao         29584 non-null  int32   
 20  dono_aceita_troca        21922 non-null  category
 21  veiculo_único_dono       10423 non-null  category
 22  revisoes_concessionaria  9172 non-null   category
 23  ipva_pago                19659 non-null  category
 24  veiculo_licenciado       15906 non-null  category
 25  garantia_de_fábrica      4365 non-null   category
 26  revisoes_dentro_agenda   5910 non-null   category
 27  veiculo_alienado         0 non-null      float64 
 28  preco                    29584 non-null  int32   
dtypes: category(19), float64(3), int16(3), int32(4)
memory usage: 3.3 MB
```

In [9]:
```
# Explorando o arquivo parte 2: Possui valores nulos?  muitos valores nulos

# Das 29.584 linhas temos: 6 colunas que serão excluida por falta de dados:
# veiculo_único_dono + revisoes_concessionaria + ipva_pago + veiculo_licenc
# garantia_de_fábrica + revisoes_dentro_agenda + veiculo_alienado + troca
# excluiremos as linhas da coluna fotos que estão com valores nulos

#df1.isnull().sum()
```

In [10]:
```
df2 = df1.drop( columns=['id','veiculo_único_dono','revisoes_concessionaria
df2 = df2.dropna(how='any', axis=0)
```

In [11]:
```
# Explorando o arquivo parte 3: Possui valores na?
#df2.isnull().sum()
```

In [12]:
```
#df2.info()
```

In [13]:
```
#df2.describe()
```

In [14]:
```
df2.agg({"preco": ["min", "mean", "max", "std", "count"]}).astype(np.int32)
```

Out[14]:

	preco
min	9869
mean	133116
max	1359812
std	81854
count	29407

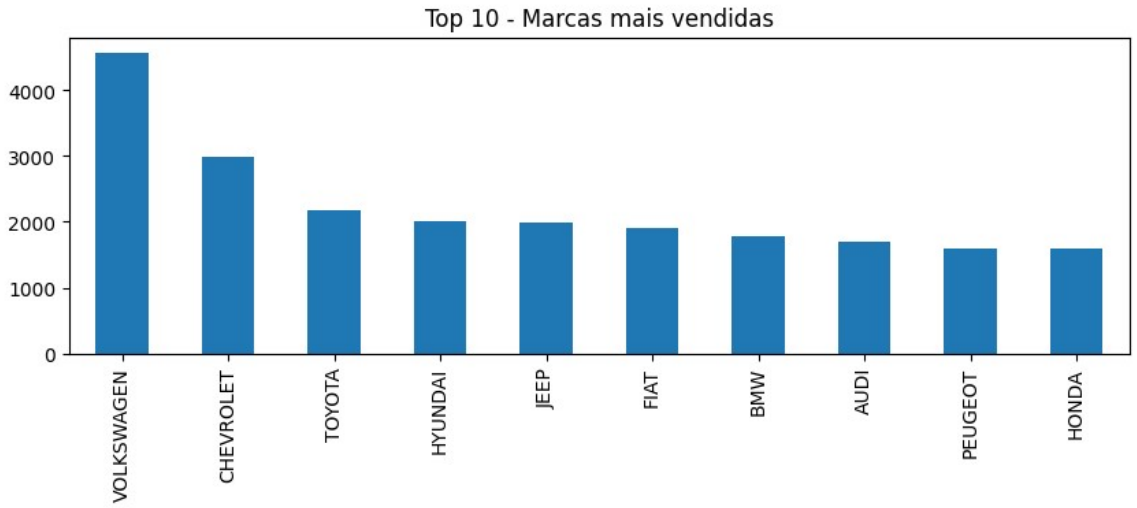
In [15]:
```
# Análise de correlação - Mapa de calor
#  Dada a alta correção e provavel efeito de colinearidade precisamos elimi
#  optamos pela coluna Ano de Fabricação.

#matriz_correlacao = df1.corr( )
#plt.figure(figsize = (10, 7))
#sns.heatmap(matriz_correlacao, annot = True, fmt='.1f')
```

In [16]:
```
df3 = df2.drop( columns=['ano_de_fabricacao'])
```

In [17]:
```
#df3.head(3)
```

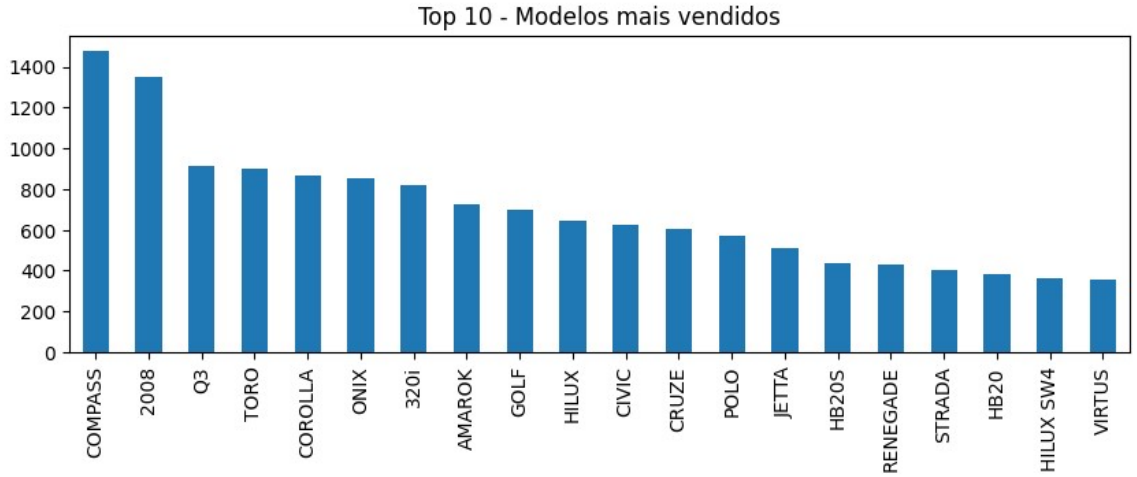
In [18]:
```
# Quais São as marcas mais vendidas? - A partir dessa perguntas descobrimos
# de automóveis deve prioriziar negociações com as 5 primeiras marcas, pois
# os veículos possuem menor tempo de permanência em estoque.

df3_Marcas_mais_vendidas = df3['marca'].value_counts()[:10]
#print(df3_Marcas_mais_vendidas)
df3_Marcas_mais_vendidas.plot(kind='bar', figsize=(10, 3))
plt.title('Top 10 - Marcas mais vendidas')
```

Out[18]: Text(0.5, 1.0, 'Top 10 - Marcas mais vendidas')



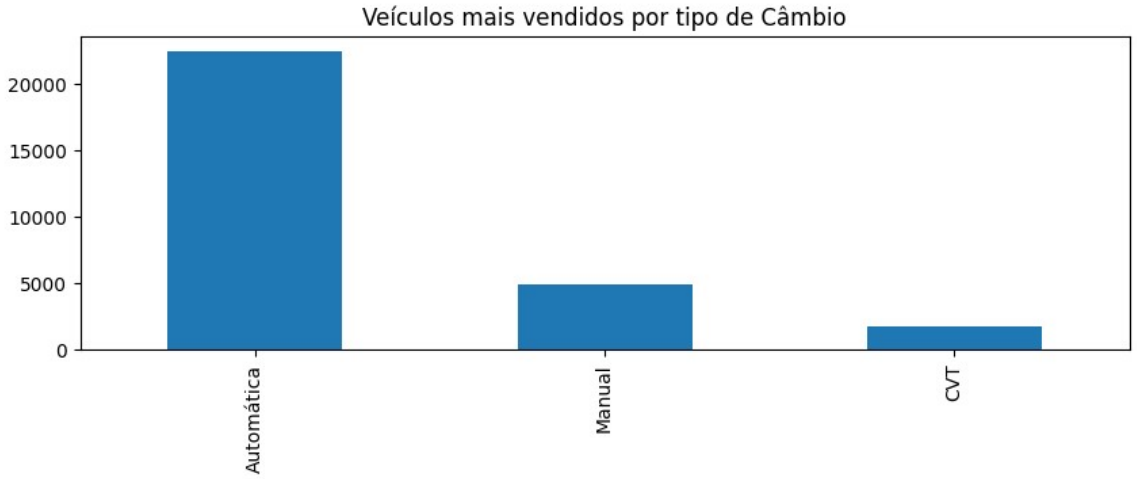
In [19]:
```
# Quais São os 10 modelos de carros mais vendidos? - A partir dessa pergunt
# de automóveis deve priorizar negociações com as 15 primeiras marcas, poi
# os veículos possuem menor tempo de permanência em estoque. Perceba que os
# ocupam as primeiras posições no top 5 modelos mais vendidos, contido ao l
# mais vendidos temos mais modelos volkswagem dentras as demais marcas.


df3_Modelos_mais_vendidos = df3['modelo'].value_counts()[:20]
#print(df3_Modelos_mais_vendidos)
df3_Modelos_mais_vendidos.plot(kind='bar', figsize=(10, 3))
plt.title('Top 10 - Modelos mais vendidos')
```

Out[19]: Text(0.5, 1.0, 'Top 10 - Modelos mais vendidos')



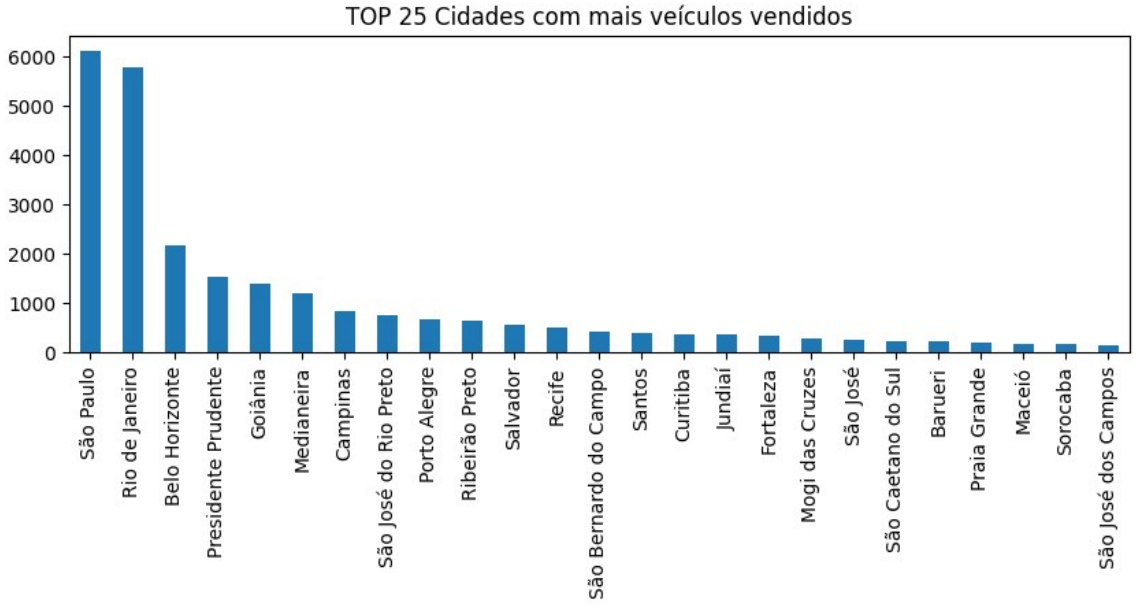
In [20]:
```
#Por esse gráfico é possível inferir que os veículos com câmbio automáticos
# da negociação. Logo deve-se dar preferência para os veículos automáticos
df3_veiculos_Cambio = df3['cambio'].value_counts()[:3]
#print(df3_veiculos_Cambio)
df3_veiculos_Cambio.plot(kind='bar', figsize=(10, 3))
plt.title('Veículos mais vendidos por tipo de Câmbio')
```

Out[20]: Text(0.5, 1.0, 'Veículos mais vendidos por tipo de Câmbio')



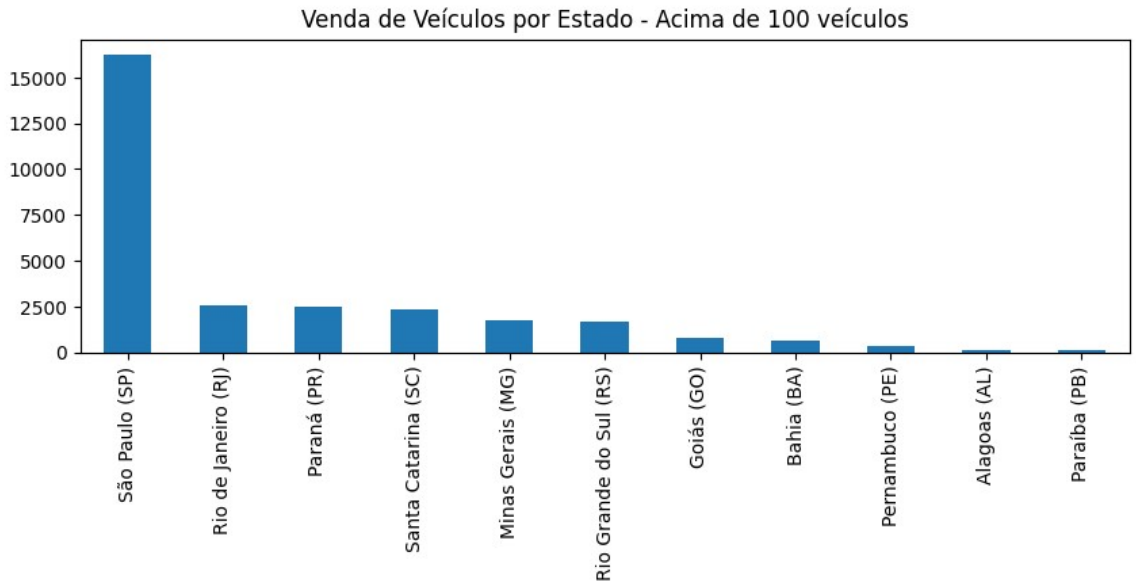
In [21]:
```
# 25 melhores cidades para vender veículos.

def3_cidade_mais_vendas = df3['cidade_vendedor'].value_counts()[:25]
#print(def3_cidade_mais_vendas)
def3_cidade_mais_vendas.plot(kind='bar', figsize=(10, 3))
plt.title('TOP 25 Cidades com mais veículos vendidos')
```

Out[21]: Text(0.5, 1.0, 'TOP 25 Cidades com mais veículos vendidos')



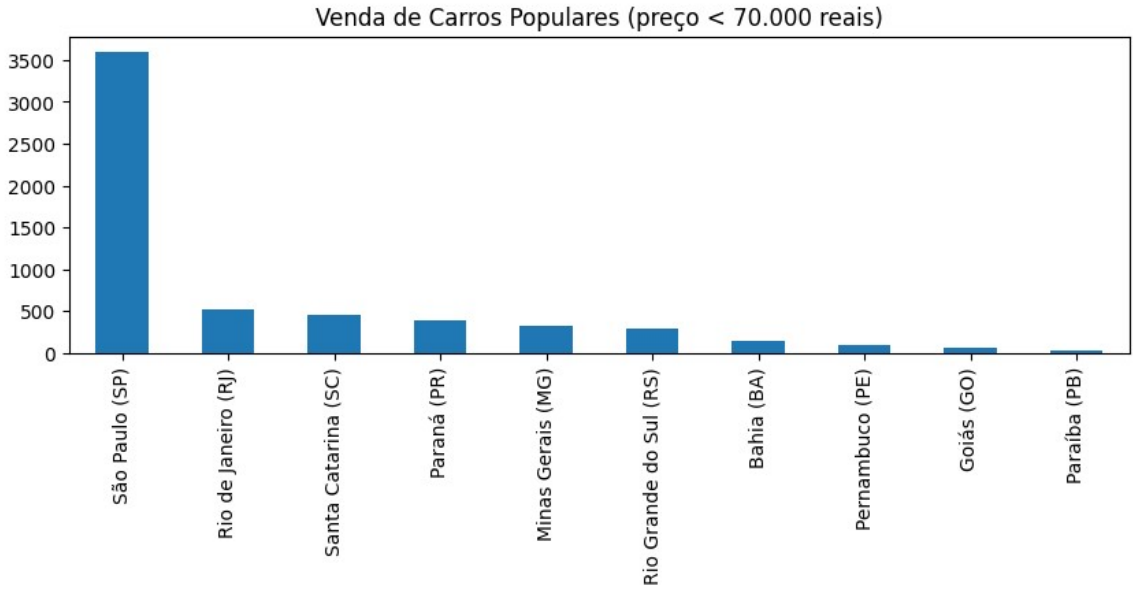
In [22]:
```
# Estados com quantidade de vendas superior a 100 veículos.

def3_estado_melhor_venda = df3['estado_vendedor'].value_counts()[:11]
#print(def3_estado_melhor_venda)
def3_estado_melhor_venda.plot(kind='bar', figsize=(10, 3))
plt.title('Venda de Veículos por Estado - Acima de 100 veículos')
```

Out[22]: Text(0.5, 1.0, 'Venda de Veículos por Estado - Acima de 100 veículos')



In [23]:
```
# Venda de Carros populares

#Qual o melhor estado cadastrado na base de dados para se vender um carro d
# O melhor estado para vender carros populares ainda é São Paulo por ser um
# A partir do terceiro Estado é possível perceber que a venda de carros pop
# relação direta com o tamanho da população.

df3_carros_populares = df3[df3['preco']<70000]['estado_vendedor'].value_cou
#print(df3_carros_populares)
df3_carros_populares.plot(kind='bar', figsize=(10, 3))
plt.title('Venda de Carros Populares (preço < 70.000 reais)')
```

Out[23]: Text(0.5, 1.0, 'Venda de Carros Populares (preço < 70.000 reais)')



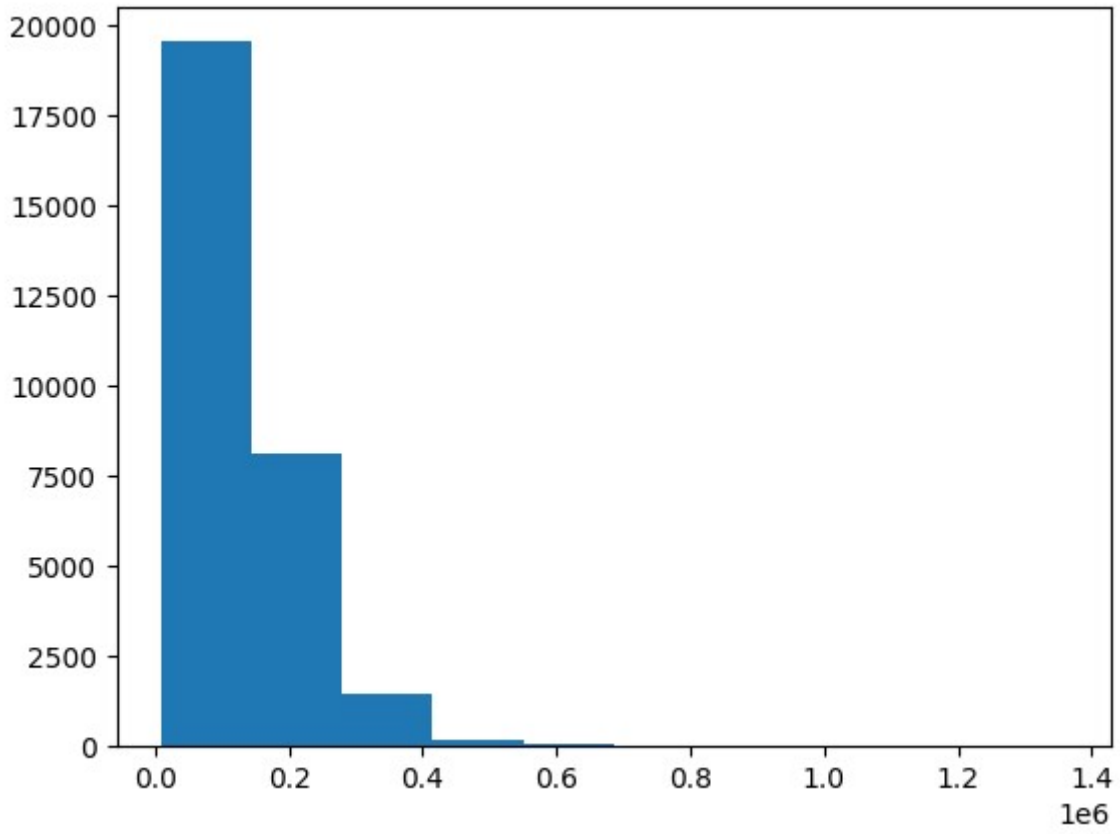
In [24]:
```
#Qual o melhor estado para se comprar uma picape com transmissão automática
# O melhor Estado para comprar uma picape com transmissão automática é São
# O segundo melhor Estado é o Paraná com 348 Picapes.

df3_Picape_automatica = df3.groupby((df3['tipo']=='Picape') & (df3['cambio'
print(df3_Picape_automatica)
```

```
False  São Paulo (SP)              14563
       Rio de Janeiro (RJ)          2222
       Paraná (PR)                  2162
       Santa Catarina (SC)          2019
       Minas Gerais (MG)            1551
       Rio Grande do Sul (RS)       1448
       Goiás (GO)                    676
       Bahia (BA)                    535
       Pernambuco (PE)               304
       Alagoas (AL)                  110
       Paraíba (PB)                  104
       Rio Grande do Norte (RN)       89
       Ceará (CE)                     69
       Pará (PA)                      55
       Amazonas (AM)                  51
       Mato Grosso do Sul (MS)        30
       Mato Grosso (MT)               25
       Acre (AC)                      23
       Espírito Santo (ES)            21
       Sergipe (SE)                   19
       Tocantins (TO)                 17
       Maranhão (MA)                   7
       Rondônia (RO)                   4
       Roraima (RR)                    2
       Piauí (PI)                      1
True   São Paulo (SP)                1712
       Paraná (PR)                    348
       Rio de Janeiro (RJ)            318
       Santa Catarina (SC)            283
       Minas Gerais (MG)              211
       Rio Grande do Sul (RS)         198
       Goiás (GO)                     102
       Bahia (BA)                      68
       Pernambuco (PE)                 14
       Alagoas (AL)                    12
       Acre (AC)                        6
       Mato Grosso (MT)                 6
       Sergipe (SE)                     5
       Mato Grosso do Sul (MS)          5
       Piauí (PI)                       4
       Paraíba (PB)                     4
       Tocantins (TO)                   3
       Rio Grande do Norte (RN)         1
       Pará (PA)                        0
       Maranhão (MA)                    0
       Rondônia (RO)                    0
       Roraima (RR)                     0
       Espírito Santo (ES)              0
       Ceará (CE)                       0
       Amazonas (AM)                    0
Name: estado_vendedor, dtype: int64
```

In [25]:
```
# Tratando outliers
```

In [26]:
```
df4 = df3
```

In [27]:
```
plt.hist(df4['preco'])
plt.show()
```



In [28]:
```
#quantidade de linhas: 29407
df4['preco'].shape
```

Out[28]: (29407,)

In [29]:
```
#outliers Superiores
df4_out = df4[df4['preco']>500000].value_counts()
df4_out.shape
```

Out[29]: (89,)

In [30]:
```
#outliers Superiores
df4_out = df4[df4['preco']<20000].value_counts()
df4_out.shape
```

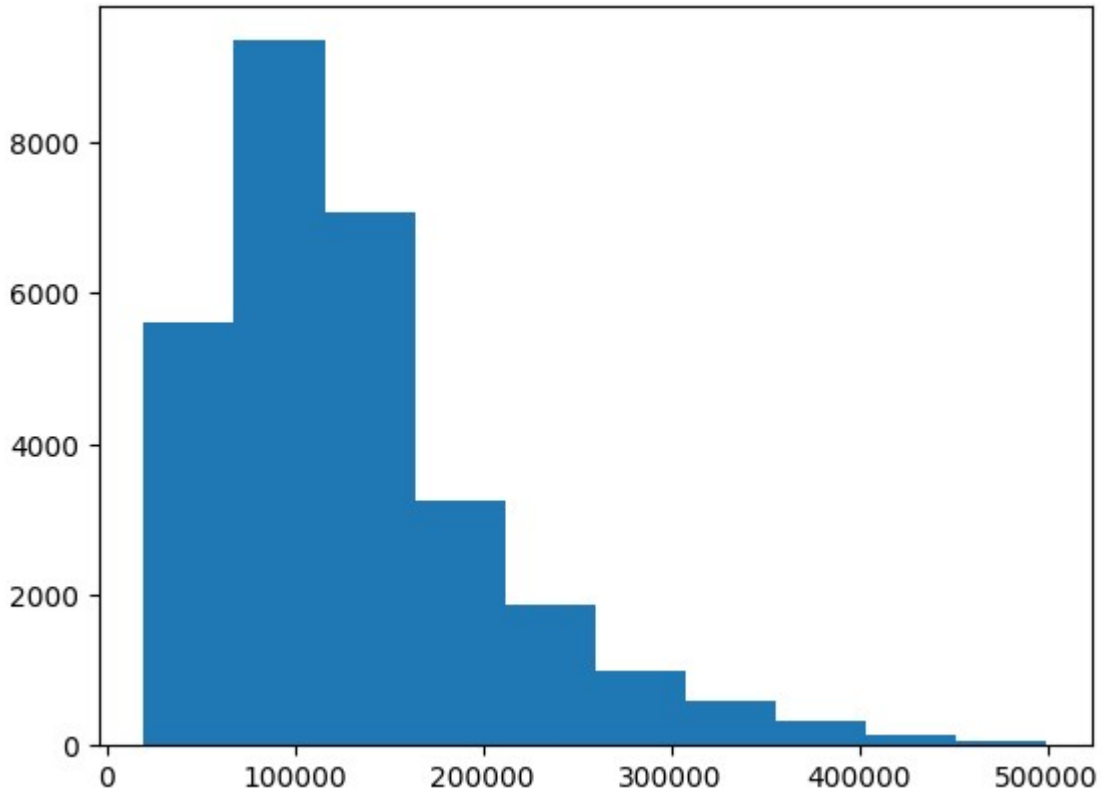
Out[30]: (28,)

In [31]:
```
# Removendo outliers
df4 = df4.loc[(df4['preco'] > 20000) & (df4['preco'] < 500000)]
```

In [32]:
```
df4.shape
```

Out[32]: (29290, 19)

In [33]:
```
plt.hist(df4['preco'])
plt.show()
```



In [85]:
```
# comecando com o df4
```

In [92]:
```
df5 = df4  # aqui troquei o 8 pelo 5
```

In [93]:
```
df5.head(2)
```

Out[93]:

	num_fotos	marca	modelo	versao	ano_modelo	hodometro	cambio	num_
0	8.0	NISSAN	KICKS	1.6 16V FLEXSTART SL 4P XTRONIC	2017	67772.0	CVT	
1	8.0	JEEP	COMPASS	2.0 16V FLEX LIMITED AUTOMÁTICO	2017	62979.0	Automática	

In [37]:
```
# Dividir as variáveis em categóricas, numéricas e a target
```

In [94]:
```
df5_target = df5['preco']
```

In [95]:
```
df5_numericas = df5[['num_fotos','ano_modelo', 'hodometro', 'num_portas', '
```

In [96]:
```
df5_categoricas = df5[['marca', 'modelo', 'versao', 'cambio', 'tipo', 'blin
```

In [97]:
```
#transformando variáveis categóricas em variáveis categóricas_codificadas.

df5_categoricas['marca_cod'] = df5_categoricas['marca'].cat.codes
df5_categoricas['modelo_cod'] = df5_categoricas['modelo'].cat.codes
df5_categoricas['versao_cod'] = df5_categoricas['versao'].cat.codes
df5_categoricas['cambio_cod'] = df5_categoricas['cambio'].cat.codes
df5_categoricas['tipo_cod'] = df5_categoricas['tipo'].cat.codes
df5_categoricas['blindado_cod'] = df5_categoricas['blindado'].cat.codes
df5_categoricas['cor_cod'] = df5_categoricas['cor'].cat.codes
df5_categoricas['tipo_vendedor_cod'] = df5_categoricas['tipo_vendedor'].cat
df5_categoricas['cidade_vendedor_cod'] = df5_categoricas['cidade_vendedor']
df5_categoricas['estado_vendedor_cod'] = df5_categoricas['estado_vendedor']
df5_categoricas['anunciante_cod'] = df5_categoricas['anunciante'].cat.codes
```

In [98]:
```
df5_categoricas_cod = df5_categoricas
```

In [99]:
```
df5_categoricas_cod = df5_categoricas.drop(['marca','modelo', 'versao', 'ca
                                            'cor', 'tipo_vendedor', 'cidade_vend
                                            'anunciante'], axis=1)
```

In [100]:
```
df5_categoricas_cod.head(3)
```

Out[100]:

	marca_cod	modelo_cod	versao_cod	cambio_cod	tipo_cod	blindado_cod	cor_cod	tipo_
0	27	257	424	4	5	0	0	
1	18	133	876	2	5	0	0	
2	19	378	1402	2	5	0	4	

In [52]:
```
# Unindo as tabelas numéricas e categóricas codificadas
```

In [101]:
```
df5_categoricas_cod.head(2)
```

Out[101]:

	marca_cod	modelo_cod	versao_cod	cambio_cod	tipo_cod	blindado_cod	cor_cod	tipo_
0	27	257	424	4	5	0	0	
1	18	133	876	2	5	0	0	

In [102]:
```
#Unindo as tabelas categoricas e numéricas

df6 = pd.concat([df5_categoricas_cod, df5_numericas], axis=1, join = 'inner
```

In [103]:
```
df6.head(2)
```

Out[103]:

	marca_cod	modelo_cod	versao_cod	cambio_cod	tipo_cod	blindado_cod	cor_cod	tipo_
0	27	257	424	4	5	0	0	
1	18	133	876	2	5	0	0	

In [ ]:
```
# Procurando os melhores parametros
```

In [59]:
```
espaco_de_parametros = {
    'bootstrap' : [False, True],
    'max_features' : [10, 20],
    'n_estimators' : [150],   # [10, 50, 100],
    'criterion' : ['friedman_mse'],
    'max_depth' : [10, 20]
}
#{'bootstrap': True,
# 'criterion': 'friedman_mse',
# 'max_depth': 20,
# 'max_features': 10,
# 'n_estimators': 150}
```

In [60]:
```
busca = GridSearchCV(RandomForestRegressor(), espaco_de_parametros,
                     cv = KFold(n_splits = 5, shuffle = True))
```

In [106]:
```
busca.fit(df6, df5.target)
```

Out[106]:
```
GridSearchCV(cv=KFold(n_splits=5, random_state=None, shuffle=True),
             estimator=RandomForestRegressor(),
             param_grid={'bootstrap': [False, True],
                         'criterion': ['friedman_mse'], 'max_depth': [10,
20],
                         'max_features': [10, 20], 'n_estimators': [150]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [117]:
```
busca.best_params_
```

Out[117]:
```
{'bootstrap': True,
 'criterion': 'friedman_mse',
 'max_depth': 20,
 'max_features': 10,
 'n_estimators': 150}
```

In [118]:
```
resultados = pd.DataFrame(busca.cv_results_)
```

In [119]:
```
resultados.head()
```

Out[119]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_bootstrap	param_cri
0	12.302841	0.227340	0.095250	0.004295	False	friedmar
1	21.344863	0.461664	0.093840	0.003601	False	friedmar
2	31.668590	0.450844	0.292192	0.004565	False	friedmar
3	52.902863	0.561118	0.292161	0.005853	False	friedmar
4	10.038171	0.229039	0.103270	0.010737	True	friedmar

In [130]:
```
#Dividindo em treino e teste

X_train, X_test, Y_train, Y_test = train_test_split(df6, df5.target, test_s
```

In [131]:
```
# parametros = Random Forest
#{'bootstrap': True, 'max_features': 10}
#{'criterion': 'friedman_mse', 'max_depth': 20, 'n_estimators': 300} = 0.754
#estimators melhor 300 testar o min_samples_leaf
#max death melhor = 17 = 75.258
#max_feature melhor = 8      .   de 10 >>15 = 74  15> 11 = 0.7518  8 = 0.75
#min_sample_leaf: melhor= 2
model1 = RandomForestRegressor( bootstrap= True,
                                min_samples_leaf= 2,
                                n_estimators= 300,
                                max_features= 8,
                                criterion = "friedman_mse",
                                max_depth = 17
                                )  #criando o modelo

model1.fit(X_train, Y_train)      # treinando o modelo
model1.score(X_test, Y_test )     # avaliando o modelo
```

Out[131]:
```
0.7702163572479355
```

In [ ]:
```
# Normalizando os dados:
```

In [132]:
```
normalizer = MinMaxScaler(feature_range=(0, 1))
```

In [133]:
```
df6_normalized = normalizer.fit_transform(df6)
```

In [ ]:
```
#aplicando os dados normalizados no modelo:
```

In [134]:
```
#Dividindo em treino e teste

X_trainN, X_testN, Y_trainN, Y_testN = train_test_split(df6_normalized, df5
```

In [135]:
```
model_normalized = RandomForestRegressor( bootstrap= True,
                                  min_samples_leaf= 2,
                                  n_estimators= 300,
                                  max_features= 8,
                                  criterion = "friedman_mse",
                                  max_depth = 17
                                  )
model_normalized.fit(X_trainN, Y_trainN)      # treinando o modelo
model_normalized.score(X_testN, Y_testN )     # avaliando o modelo
```

Out[135]: 0.7593613780744438

In [ ]:

In [ ]:
```
# Padronizando os dados
```

In [136]:
```
standardize = StandardScaler().fit(df6)
```

In [137]:
```
df6_standardized = standardize.transform(df6)
```

In [138]:
```
#Dividindo em treino e teste

X_trainS, X_testS, Y_trainS, Y_testS = train_test_split(df6_standardized, d
```

In [ ]:
```
#aplicando os dados normalizados no modelo:
```

In [139]:
```
model_standardized = RandomForestRegressor( bootstrap= True,
                                  min_samples_leaf= 2,
                                  n_estimators= 300,
                                  max_features= 8,
                                  criterion = "friedman_mse",
                                  max_depth = 17
                                  )  #criando o modelo
model_standardized.fit(X_trainS, Y_trainS)      # treinando o modelo
model_standardized.score(X_testS, Y_testS )     # avaliando o modelo
```

Out[139]: 0.7630954871256815

In [140]:
```
df9_Standardized_dataframe = pd.DataFrame(df9_standardized)
```

In [141]:
```
df9_Standardized_dataframe.head()
```

Out[141]:

	0	1	2	3	4	5	6	7	
0	0.558272	0.237375	-0.771881	1.161686	0.462544	-0.090702	-0.588150	-0.802858	0.43
1	-0.143337	-0.726036	0.092320	-0.521884	0.462544	-0.090702	-0.588150	-0.802858	-2.01
2	-0.065381	1.177478	1.098004	-0.521884	0.462544	-0.090702	1.926637	1.245550	0.66
3	1.415794	-1.269897	0.495741	-0.521884	-0.178050	-0.090702	-0.588150	1.245550	0.76
4	1.026011	0.252914	0.474709	-0.521884	1.103138	-0.090702	1.926637	-0.802858	0.43

In [ ]:

In [ ]:

In [ ]:

In [ ]: