Clase02Estadistica

November 18, 2024

Librerias

[1]:
```python
import os
os.chdir('C://Users//jaag2//OneDrive//PuraVidaAnalytics//00 - Clientes//03 -␣
↪Texas Tech//07 - Cursos//01 - No Supervisado//03 - Estadistica')
```

[2]:
```python
import pandas as pd
import numpy as np
```

1 Leer Datos

[3]:
```python
carsdf = pd.read_csv("cars.csv")
carsdf
```

[3]:
```
      manufacturer_name  model_name transmission   color  odometer_value  \
0                Subaru     Outback    automatic  silver          190000
1                Subaru     Outback    automatic    blue          290000
2                Subaru    Forester    automatic     red          402000
3                Subaru     Impreza   mechanical    blue           10000
4                Subaru      Legacy    automatic   black          280000
…                   …          …           …       …               …
38526          Chrysler         300    automatic  silver          290000
38527          Chrysler  PT Cruiser   mechanical    blue          321000
38528          Chrysler         300    automatic    blue          777957
38529          Chrysler  PT Cruiser   mechanical   black           20000
38530          Chrysler     Voyager    automatic  silver          297729

       year_produced engine_fuel  engine_has_gas engine_type  engine_capacity  \
0               2010    gasoline           False    gasoline              2.5
1               2002    gasoline           False    gasoline              3.0
2               2001    gasoline           False    gasoline              2.5
3               1999    gasoline           False    gasoline              3.0
4               2001    gasoline           False    gasoline              2.5
…                  …          …               …          …                …
38526           2000    gasoline           False    gasoline              3.5
38527           2004      diesel           False      diesel              2.2
38528           2000    gasoline           False    gasoline              3.5
```

```
38529          2001    gasoline          False    gasoline             2.0
38530          2000    gasoline          False    gasoline             2.4

       … feature_1  feature_2 feature_3 feature_4  feature_5  feature_6  \
0      …      True       True      True     False       True      False
1      …      True      False     False      True       True      False
2      …      True      False     False     False      False      False
3      …     False      False     False     False      False      False
4      …      True      False      True      True      False      False
…    …       …        …       …       …       …        …
38526  …      True      False     False      True       True      False
38527  …      True      False     False      True       True      False
38528  …      True      False     False      True       True      False
38529  …      True      False     False     False      False      False
38530  …     False      False     False     False      False      False

      feature_7  feature_8  feature_9  duration_listed
0          True       True       True               16
1         False      False       True               83
2         False       True       True              151
3         False      False      False               86
4         False      False       True                7
…           …        …        …              …
38526     False       True       True              301
38527     False       True       True              317
38528     False       True       True              369
38529     False      False       True              490
38530     False      False       True              632

[38531 rows x 30 columns]
```

[4]:
```
carsdf.dtypes # Tipos de Datos
```

[4]:
```
manufacturer_name     object
model_name            object
transmission          object
color                 object
odometer_value         int64
year_produced          int64
engine_fuel           object
engine_has_gas          bool
engine_type           object
engine_capacity      float64
body_type             object
has_warranty            bool
state                 object
drivetrain            object
```

```
price_usd            float64
is_exchangeable         bool
location_region       object
number_of_photos       int64
up_counter             int64
feature_0               bool
feature_1               bool
feature_2               bool
feature_3               bool
feature_4               bool
feature_5               bool
feature_6               bool
feature_7               bool
feature_8               bool
feature_9               bool
duration_listed        int64
dtype: object
```

Generar un conjunto de estadísticos descriptivos del dataset:

[5]:
```
carsdf.describe()
```

[5]:
```
       odometer_value  year_produced  engine_capacity     price_usd  \
count    38531.000000   38531.000000     38521.000000  38531.000000
mean    248864.638447    2002.943734         2.055161   6639.971021
std     136072.376530       8.065731         0.671178   6428.152018
min          0.000000    1942.000000         0.200000      1.000000
25%     158000.000000    1998.000000         1.600000   2100.000000
50%     250000.000000    2003.000000         2.000000   4800.000000
75%     325000.000000    2009.000000         2.300000   8990.000000
max    1000000.000000    2019.000000         8.000000  50000.000000

       number_of_photos    up_counter  duration_listed
count      38531.000000  38531.000000     38531.000000
mean           9.649062     16.306091        80.577249
std            6.093217     43.286933       112.826569
min            1.000000      1.000000         0.000000
25%            5.000000      2.000000        23.000000
50%            8.000000      5.000000        59.000000
75%           12.000000     16.000000        91.000000
max           86.000000   1861.000000      2232.000000
```

2 Medidas de Tendencia Central

[6]:
```
carsdf.price_usd.mean()
```

[6]: 6639.971021255613

[7]:
```
carsdf['price_usd'].mean()
```

[7]: 6639.971021255613

[8]:
```
carsdf['price_usd'].quantile(0.25)
```

[8]: 2100.0

[9]:
```
carsdf['price_usd'].std()
```

[9]: 6428.152018202915

[10]:
```
carsdf['price_usd'].quantile(.50) #Mediana
```

[10]: 4800.0

[11]:
```
carsdf['price_usd'].min()
```

[11]: 1.0

[12]:
```
carsdf['price_usd'].max()
```

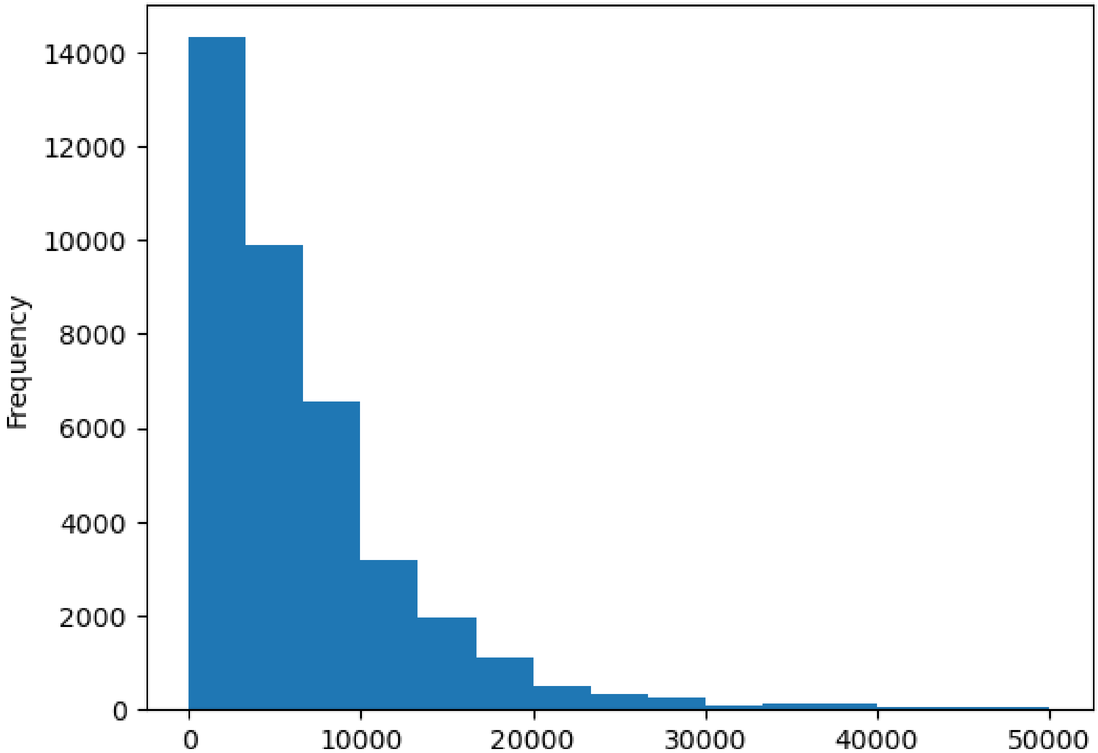
[12]: 50000.0

[13]:
```
carsdf['price_usd'].mode()
```

[13]:
```
0    1500.0
Name: price_usd, dtype: float64
```

[14]:
```
carsdf['price_usd'].plot.hist(bins=15)
```

[14]: <Axes: ylabel='Frequency'>

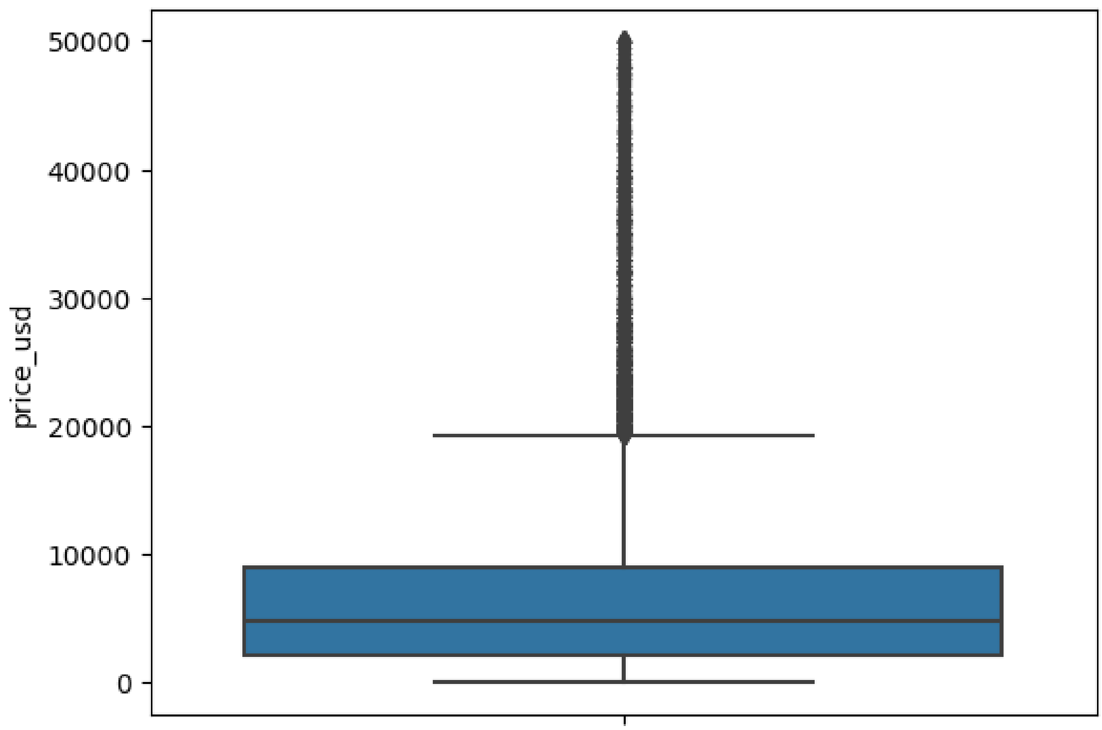


[15]:
```
#Rango = valor max - valor min
rango = carsdf['price_usd'].max() - carsdf['price_usd'].min()
rango
```

[15]: 49999.0

[16]:
```
import seaborn as sns
sns.boxplot(
    y     = 'price_usd',
    data  = carsdf
)
```

[16]: <Axes: ylabel='price_usd'>



[17]:
```
#Quartiles
median = carsdf['price_usd'].median()
Q1 = carsdf['price_usd'].quantile(q=0.25) #toma el primer 25% de todos los datos
Q3 = carsdf['price_usd'].quantile(q=0.75)
min_val = carsdf['price_usd'].quantile(q=0)
max_val = carsdf['price_usd'].quantile(q=1)
print(min_val, Q1, median, Q3, max_val)
```

```
1.0 2100.0 4800.0 8990.0 50000.0
```

[18]:
```
iqr = Q3 - Q1
iqr
```

[18]: 6890.0

2.1 Límites para detección de outliers (con datos simétricamente distribuídos)

[19]:
```
minlimit = Q1 - 1.5*iqr
maxlimit = Q3 + 1.5*iqr
print(minlimit, maxlimit)
```

```
-8235.0 19325.0
```

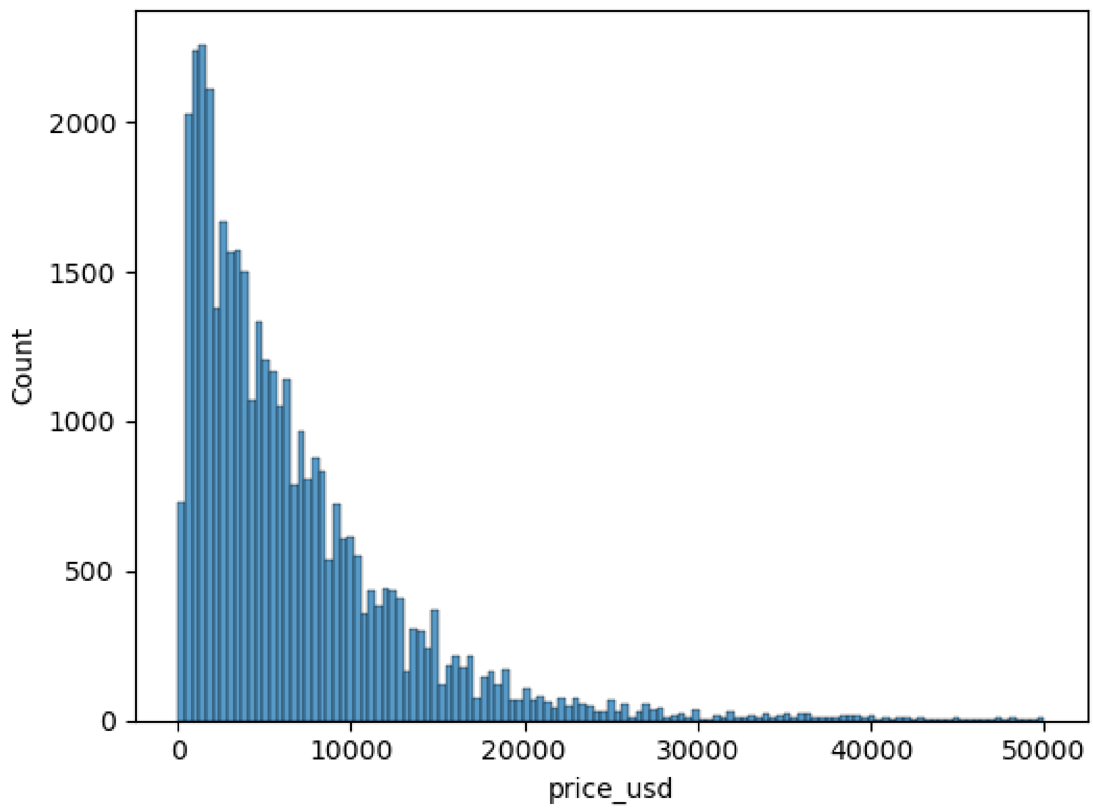
[20]:
```
minlimit
```

[20]: -8235.0

[21]:
```
sns.histplot(carsdf['price_usd'])
```

```
c:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

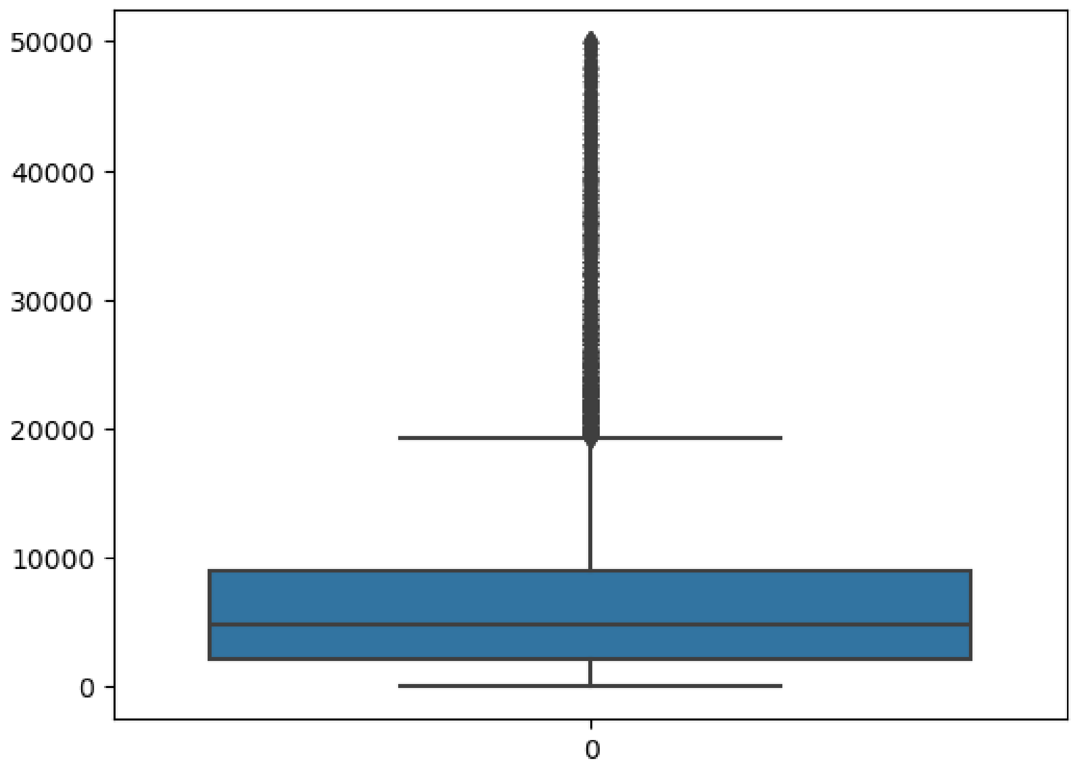
[21]: <Axes: xlabel='price_usd', ylabel='Count'>



Nota 1: El valor es negativo porque se está aplicando una ecuación de una distribución simética a una no simétrica.

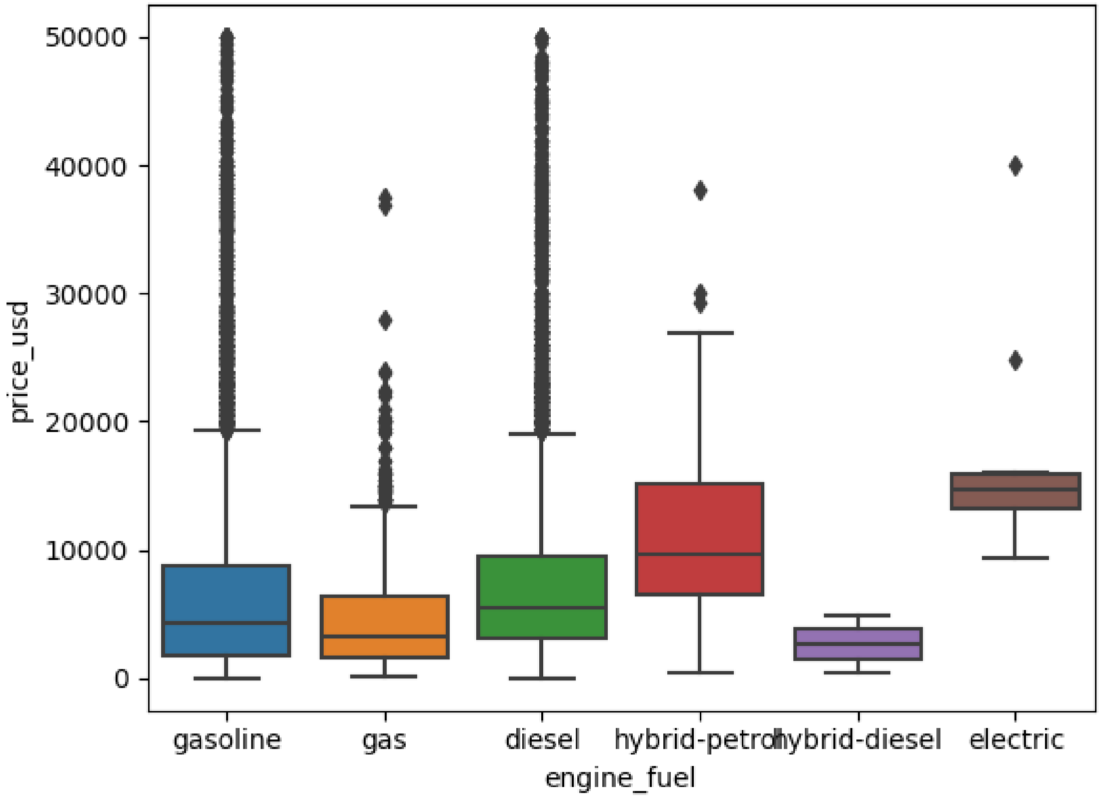
[22]:
```
sns.boxplot(carsdf['price_usd'])
```

[22]: <Axes: >




```
[23]: sns.boxplot(x='engine_fuel', y='price_usd', data=carsdf)
```

```
[23]: <Axes: xlabel='engine_fuel', ylabel='price_usd'>
```



[24]:
```
carsdf.dtypes
```

[24]:
```
manufacturer_name     object
model_name            object
transmission          object
color                 object
odometer_value         int64
year_produced          int64
engine_fuel           object
engine_has_gas          bool
engine_type           object
engine_capacity      float64
body_type             object
has_warranty            bool
state                 object
drivetrain            object
price_usd            float64
is_exchangeable         bool
location_region       object
number_of_photos       int64
up_counter             int64
```

```
feature_0                  bool
feature_1                  bool
feature_2                  bool
feature_3                  bool
feature_4                  bool
feature_5                  bool
feature_6                  bool
feature_7                  bool
feature_8                  bool
feature_9                  bool
duration_listed           int64
dtype: object
```

[25]:
```
carsdf.select_dtypes(include=['float64', 'int64'])
```

[25]:
```
       odometer_value  year_produced  engine_capacity  price_usd  \
0              190000           2010              2.5   10900.00
1              290000           2002              3.0    5000.00
2              402000           2001              2.5    2800.00
3               10000           1999              3.0    9999.00
4              280000           2001              2.5    2134.11
…                 …              …                …         …
38526          290000           2000              3.5    2750.00
38527          321000           2004              2.2    4800.00
38528          777957           2000              3.5    4300.00
38529           20000           2001              2.0    4000.00
38530          297729           2000              2.4    3200.00

       number_of_photos  up_counter  duration_listed
0                     9          13               16
1                    12          54               83
2                     4          72              151
3                     9          42               86
4                    14           7                7
…                   …           …              …
38526                 5          85              301
38527                 4          20              317
38528                 3          63              369
38529                 7         156              490
38530                 4          73              632

[38531 rows x 7 columns]
```

[26]:
```
corr_matrix = carsdf.select_dtypes(include=['float64', 'int64']).
 ↪corr(method='pearson')
```

[27]:
```
corr_matrix
```

[27]:

```
                   odometer_value  year_produced  engine_capacity  price_usd  \
odometer_value           1.000000      -0.488679         0.105704  -0.421204
year_produced           -0.488679       1.000000         0.005059   0.705511
engine_capacity          0.105704       0.005059         1.000000   0.296597
price_usd               -0.421204       0.705511         0.296597   1.000000
number_of_photos        -0.143708       0.258180         0.106691   0.316859
up_counter              -0.020961       0.007945         0.079152   0.057382
duration_listed         -0.000428      -0.017001         0.080081   0.033524

                   number_of_photos  up_counter  duration_listed
odometer_value            -0.143708   -0.020961        -0.000428
year_produced              0.258180    0.007945        -0.017001
engine_capacity            0.106691    0.079152         0.080081
price_usd                  0.316859    0.057382         0.033524
number_of_photos           1.000000    0.073891        -0.028255
up_counter                 0.073891    1.000000         0.698116
duration_listed           -0.028255    0.698116         1.000000
```

[28]:

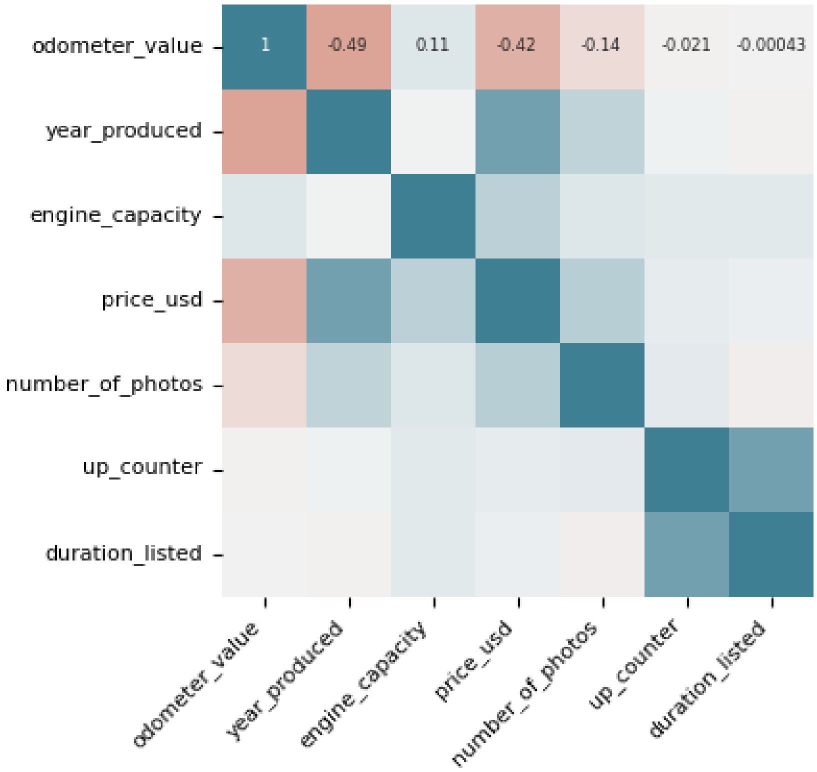
```python
import matplotlib.pyplot as plt
```

[29]:

```python
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(4, 4))

sns.heatmap(
    corr_matrix,
    annot     = True,
    cbar      = False,
    annot_kws = {"size": 6},
    vmin      = -1,
    vmax      = 1,
    center    = 0,
    cmap      = sns.diverging_palette(20, 220, n=200),
    square    = True,
    ax        = ax
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation = 45,
    horizontalalignment = 'right',
)

ax.tick_params(labelsize = 8)
```



3 Tipos de escalamiento

3.1 Lineal

Min-max: hace una transformación para que los datos entren en el rango [-1, 1] o [0,1] por medio de una fórmula. Es uno de los más usados. Funciona mejor para datos uniformemente distribuidos.

Z-Score: es uno de los más comunes porque está basado en el promedio y desviación estándar. Funciona mejor para datos distribuidos "normalmente" (forma de campana de Gauss).

[30]:
```
X = carsdf['price_usd']
```

[31]:
```
X
```

[31]:
```
0        10900.00
1         5000.00
2         2800.00
3         9999.00
4         2134.11
           …
38526     2750.00
38527     4800.00
```

```
38528     4300.00
38529     4000.00
38530     3200.00
Name: price_usd, Length: 38531, dtype: float64
```

[32]:
```python
Z_X = (X - X.mean())/X.std()
```

[33]:
```python
Z_X
```

[33]:
```
0         0.662714
1        -0.255123
2        -0.597368
3         0.522550
4        -0.700957
            …
38526    -0.605146
38527    -0.286236
38528    -0.364019
38529    -0.410689
38530    -0.535142
Name: price_usd, Length: 38531, dtype: float64
```

[34]:
```python
Z_X.mean()
```

[34]: -6.78621698787254e-17

[35]:
```python
Z_X.std()
```

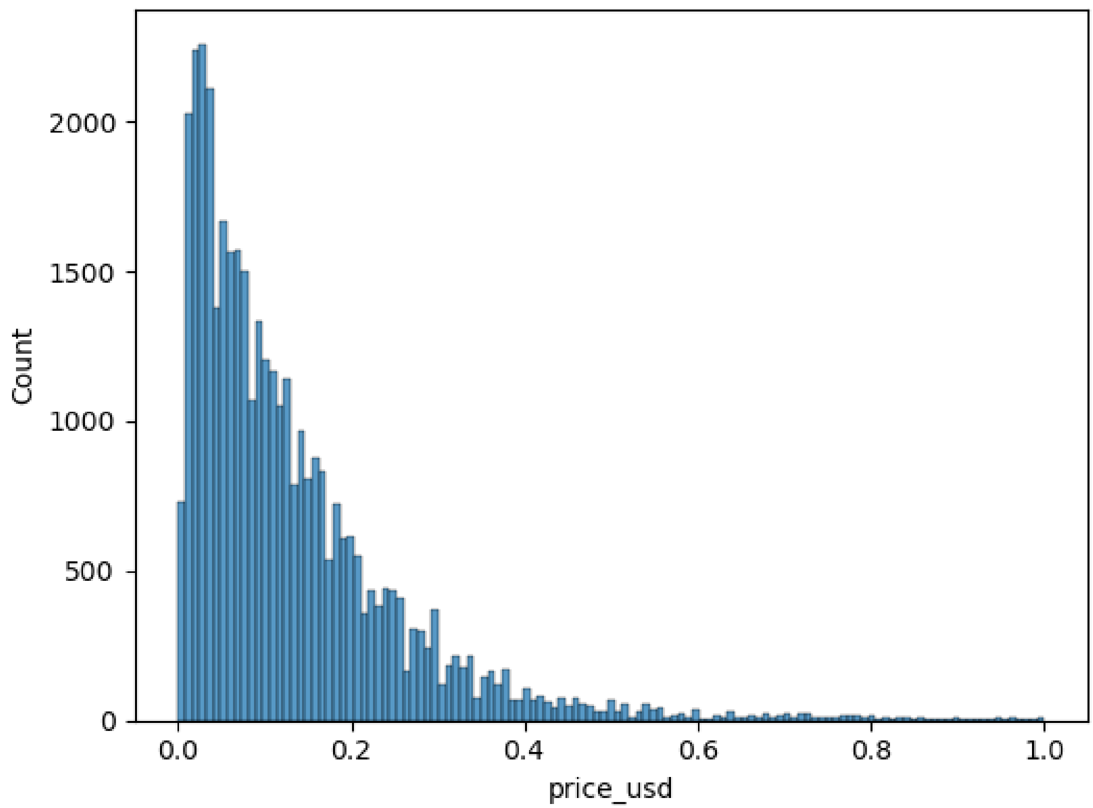
[35]: 1.000000000000016

[36]:
```python
MM_X = (X-X.min())/(X.max() - X.min())
```

[37]:
```python
sns.histplot(MM_X)
```

```
c:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

[37]: <Axes: xlabel='price_usd', ylabel='Count'>

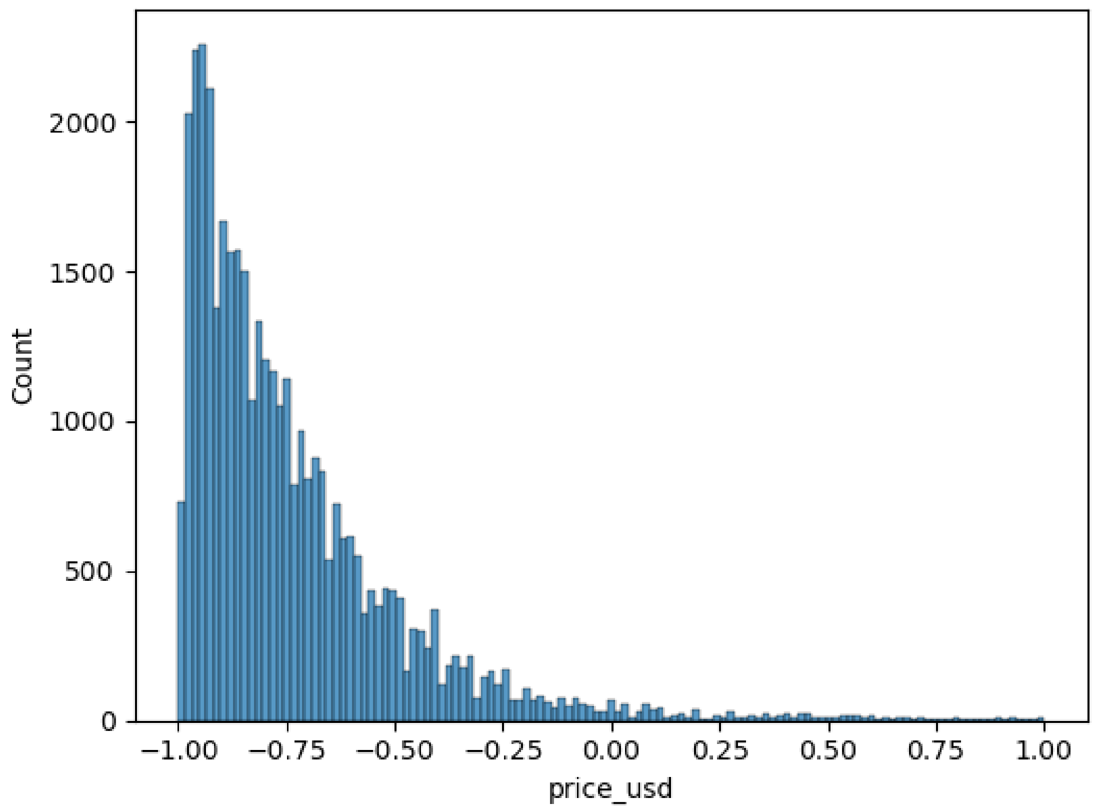


[38]:
```
MM2_X = (2*X-X.min()-X.max())/(X.max() - X.min())
```

[39]:
```
sns.histplot(MM2_X)
```

```
c:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

[39]: <Axes: xlabel='price_usd', ylabel='Count'>



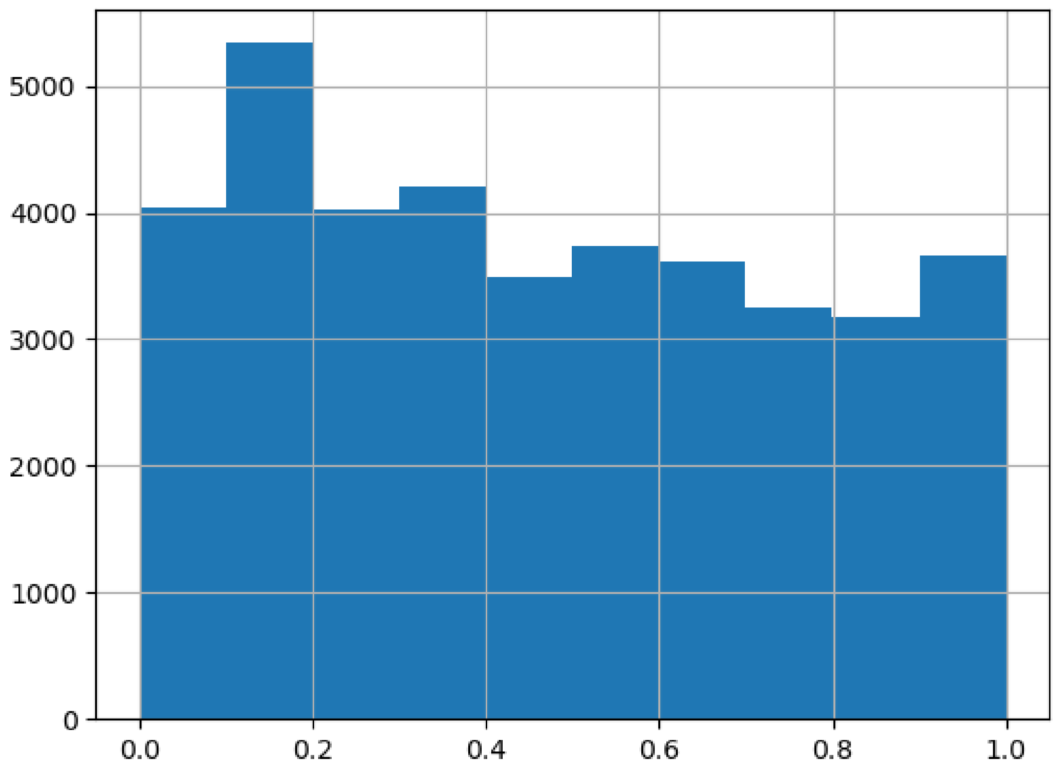
3.2 Transformaciones no lineales en Python

[40]:
```
X_es = X/10000
```

[41]:
```
X_tan = np.tanh(X_es)
```

[42]:
```
X_tan.hist()
```

[42]: <Axes: >



4 Procesamiento para variables categóricas

Dummy: es la representación más compacta que se puede tener de los datos. Es mejor usarla cuando los inputs son variables linealmente independientes (no tienen un grado de correlación significativo). Es decir, las cuando se sabe que las categorías son independientes entre sí. **One-hot:** es más general. Permite incluir categorías que no estaban en el dataset inicialmente. De forma que si se filtra una categoría que no estaba incluida, igual se pueda representar numéricamente.

[43]:
```
ZZ = pd.get_dummies(carsdf['engine_type'])
```

[44]:
```
ZZ
```

[44]:

	diesel	electric	gasoline
0	False	False	True
1	False	False	True
2	False	False	True
3	False	False	True
4	False	False	True
…	…	…	…
38526	False	False	True
38527	True	False	False
38528	False	False	True

```
38529    False     False      True
38530    False     False      True

[38531 rows x 3 columns]
```

[45]:
```python
import sklearn.preprocessing as preprocessing

encoder = preprocessing.OneHotEncoder(handle_unknown='ignore')
```

[46]:
```python
tmp = carsdf['engine_type'].unique()
```

[47]:
```python
tmp
```

[47]:
```
array(['gasoline', 'diesel', 'electric'], dtype=object)
```

[48]:
```python
encoder.fit(carsdf[['engine_type']].values)
```

[48]:
```
OneHotEncoder(handle_unknown='ignore')
```

[49]:
```python
encoder
```

[49]:
```
OneHotEncoder(handle_unknown='ignore')
```

[50]:
```python
encoder.transform([['gasoline'],['diesel'], ['aceite'],['agua']]).toarray()
```

[50]:
```
array([[0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

[51]:
```python
encoder.transform([['gasoline'],['diesel'], ['aceite']]).toarray()
```

[51]:
```
array([[0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 0.]])
```

5 Álgebra Lineal aplicada a la matriz de correlaciones

[52]:
```python
corr_matrix
```

[52]:
```
                  odometer_value  year_produced  engine_capacity  price_usd  \
odometer_value          1.000000      -0.488679         0.105704  -0.421204
year_produced          -0.488679       1.000000         0.005059   0.705511
engine_capacity         0.105704       0.005059         1.000000   0.296597
price_usd              -0.421204       0.705511         0.296597   1.000000
number_of_photos       -0.143708       0.258180         0.106691   0.316859
```

```
up_counter             -0.020961       0.007945         0.079152    0.057382
duration_listed        -0.000428      -0.017001         0.080081    0.033524

                  number_of_photos  up_counter  duration_listed
odometer_value           -0.143708   -0.020961        -0.000428
year_produced             0.258180    0.007945        -0.017001
engine_capacity           0.106691    0.079152         0.080081
price_usd                 0.316859    0.057382         0.033524
number_of_photos          1.000000    0.073891        -0.028255
up_counter                0.073891    1.000000         0.698116
duration_listed          -0.028255    0.698116         1.000000
```

[53]:
```python
from numpy.linalg import eig
```

[54]:
```python
w,v=eig(corr_matrix)
```

[55]:
```python
w
```

[55]:
```
array([2.26276437, 1.71228239, 1.12619392, 0.83873606, 0.53102881,
       0.23486557, 0.29412887])
```

[56]:
```python
w.sum()
```

[56]:
```
7.000000000000007
```

[57]:
```python
w # valores propios
```

[57]:
```
array([2.26276437, 1.71228239, 1.12619392, 0.83873606, 0.53102881,
       0.23486557, 0.29412887])
```

[58]:
```python
w
```

[58]:
```
array([2.26276437, 1.71228239, 1.12619392, 0.83873606, 0.53102881,
       0.23486557, 0.29412887])
```

[59]:
```python
porc_varianza = w/w.sum()
```

[60]:
```python
porc_varianza
```

[60]:
```
array([0.32325205, 0.24461177, 0.16088485, 0.11981944, 0.07586126,
       0.03355222, 0.04201841])
```

[61]:
```python
porc_varianza
```

[61]:
```
array([0.32325205, 0.24461177, 0.16088485, 0.11981944, 0.07586126,
       0.03355222, 0.04201841])
```

[62]:
```
porc_varianza.sum()
```

[62]:
```
0.9999999999999999
```

[63]:
```
V_pd = pd.DataFrame.from_records(v)
```

[64]:
```
V_pd.T.dot(V_pd)
```

[64]:
```
              0             1             2             3             4  \
0  1.000000e+00 -1.352229e-16 -3.328643e-16  8.855268e-17  6.918030e-17
1 -1.352229e-16  1.000000e+00  4.073029e-16 -6.527008e-17 -6.617097e-16
2 -3.328643e-16  4.073029e-16  1.000000e+00  1.627564e-16  1.207682e-16
3  8.855268e-17 -6.527008e-17  1.627564e-16  1.000000e+00 -7.056217e-16
4  6.918030e-17 -6.617097e-16  1.207682e-16 -7.056217e-16  1.000000e+00
5  9.916572e-17  1.877511e-16 -8.315955e-17 -1.371748e-16 -3.246695e-16
6  5.706270e-17 -4.578607e-16 -1.785732e-16  5.362632e-16  8.951776e-17

              5             6
0  9.916572e-17  5.706270e-17
1  1.877511e-16 -4.578607e-16
2 -8.315955e-17 -1.785732e-16
3 -1.371748e-16  5.362632e-16
4 -3.246695e-16  8.951776e-17
5  1.000000e+00  9.615936e-16
6  9.615936e-16  1.000000e+00
```