```
import xgboost as xgb
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import f1_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import root_mean_squared_error
from bayes_opt import BayesianOptimization
```

```
#df = pd.read_csv('../data/sample submission.csv')
df = pd.read_csv('../data/data.csv', sep=';', encoding='latin1')
print(df)
```

```
          Id     Category   Manufacturer     Model  Prod. year Gear box type  \
0       2680         Jeep        HYUNDAI        H1        2014     Automatic   
1       5960        Sedan     MITSUBISHI    Mirage        2002     Automatic   
2       2185         Jeep        HYUNDAI  Santa FE        2014     Automatic   
3      15905        Sedan  MERCEDES-BENZ     E 260        1992        Manual   
4      15337    Universal          HONDA       FIT        2015     Automatic   
...      ...          ...            ...       ...         ...           ...   
16346  19198         Jeep         TOYOTA     RAV 4        2015     Automatic   
16347   3583        Sedan         TOYOTA     Prius        2009     Automatic   
16348  18497         Jeep      SSANGYONG    REXTON        2015     Automatic   
16349   4565  Goods wagon           OPEL     Combo        2011        Manual   
16350  11586        Sedan           FORD    Fusion        2013     Automatic   

      Leather interior Fuel type Engine volume Drive wheels  Cylinders  \
0                  Yes    Diesel           2.5        Front          4   
1                   No    Petrol           1.8        Front          4   
2                  Yes    Diesel             2        Front          4   
3                   No       CNG           2.6         Rear          6   
4                  Yes    Hybrid           1.5        Front          4   
...                ...       ...           ...          ...        ...   
16346              Yes    Petrol           2.5          4x4          4   
16347              Yes    Hybrid           1.5        Front          4   
16348              Yes    Diesel             2        Front          4   
16349               No    Diesel     1.3 Turbo        Front          4   
16350              Yes    Hybrid             2        Front          4   

         Mileage Doors  Airbags       Wheel   Color Sales Fee  price  
0       74210 km     4        4  Left wheel  Silver       777  22433  
1      160000 km     4        2  Left wheel   White         -   7500  
2       51106 km     4        4  Left wheel   White       639  27284  
3           0 km     4        4  Left wheel   Beige         -   3450  
4       35624 km     4        4  Left wheel   Black       308  26644  
...          ...   ...      ...         ...     ...       ...    ...  
16346  149019 km     4        0  Left wheel    Grey       934  28225  
16347  142426 km     4       12  Left wheel   White       746   1882  
16348  123303 km     4        4  Left wheel   Black       765  36219  
16349   95000 km     4        4  Left wheel   White       490   9408  
```

```
16350  174619 km     4        0  Left wheel    Grey       640   1646

[16351 rows x 18 columns]
```

## DATOS FALTANTES

```
# verificar datos faltantes
for col in df.columns.to_list():
  calc = (df[col].isna().sum()/df.shape[0])*100
  print(f'{col} missing Values: {calc}%')
```

```
Id missing Values: 0.0%
Category missing Values: 0.0%
Manufacturer missing Values: 0.0%
Model missing Values: 0.0%
Prod. year missing Values: 0.0%
Gear box type missing Values: 0.0%
Leather interior missing Values: 0.0%
Fuel type missing Values: 0.0%
Engine volume missing Values: 0.0%
Drive wheels missing Values: 0.0%
Cylinders missing Values: 0.0%
Mileage missing Values: 0.0%
Doors missing Values: 0.0%
Airbags missing Values: 0.0%
Wheel missing Values: 0.0%
Color missing Values: 0.0%
Sales Fee missing Values: 0.0%
price missing Values: 0.0%
```

## VARIABLES CATEGÓRICAS

## ENCODING

```
def label_encoding(dataset, column_name):
    label_encoder = LabelEncoder()
    dataset[column_name] = label_encoder.fit_transform(dataset[column_name])
    return dataset, label_encoder

def frequency_encoding(dataset, col):
    freq = dataset[col].value_counts(normalize=True)
    dataset[col] = dataset[col].map(freq)
    return dataset, freq
```

```
df2 = df
def to_zero(n):
  if n == '-': return 0
  return n
```

```
def mileage_km(n):
  return n.replace(' km', '')

def turbo(n):
  if 'Turbo' in n: return 1
  return 0

def engine_volume(n):
  return n.replace(' Turbo', '')

def doors(n):
  if n == '>5': return 6
  return n


df2['Turbo'] = df2['Engine volume'].map(turbo)



df2['Sales Fee'] = df2['Sales Fee'].map(to_zero)
df2['Mileage'] = df2['Mileage'].map(mileage_km)
df2['Engine volume'] = df2['Engine volume'].map(engine_volume)
df2['Doors'] = df2['Doors'].map(doors)

df2.head(20)
```

| | Id | Category | Manufacturer | Model | Prod. year | Gear box type | Leather interior | Fuel type | Engine volume | Drive wheels | Cylinders | Mileage | Doors | Airbags | Wheel | Color | Sales Fee | price | Turbo |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2680 | Jeep | HYUNDAI | H1 | 2014 | Automatic | Yes | Diesel | 2.5 | Front | 4 | 74210 | 4 | 4 | Left wheel | Silver | 777 | 22433 | 0 |
| 1 | 5960 | Sedan | MITSUBISHI | Mirage | 2002 | Automatic | No | Petrol | 1.8 | Front | 4 | 160000 | 4 | 2 | Left wheel | White | 0 | 7500 | 0 |
| 2 | 2185 | Jeep | HYUNDAI | Santa FE | 2014 | Automatic | Yes | Diesel | 2 | Front | 4 | 51106 | 4 | 4 | Left wheel | White | 639 | 27284 | 0 |
| 3 | 15905 | Sedan | MERCEDES-BENZ | E 260 | 1992 | Manual | No | CNG | 2.6 | Rear | 6 | 0 | 4 | 4 | Left wheel | Beige | 0 | 3450 | 0 |
| 4 | 15337 | Universal | HONDA | FIT | 2015 | Automatic | Yes | Hybrid | 1.5 | Front | 4 | 35624 | 4 | 4 | Left wheel | Black | 308 | 26644 | 0 |
| 5 | 13792 | Hatchback | HONDA | FIT | 2014 | Automatic | Yes | Petrol | 1.5 | Front | 4 | 78000 | 4 | 4 | Left wheel | White | 501 | 25638 | 0 |
| 6 | 12015 | Microbus | FORD | Transit | 2007 | Manual | No | Diesel | 2.4 | Rear | 4 | 165000 | 4 | 2 | Left wheel | Blue | 0 | 17249 | 0 |
| 7 | 307 | Sedan | TOYOTA | Camry | 2015 | Automatic | Yes | Hybrid | 2.5 | Front | 4 | 35000 | 4 | 10 | Left wheel | Grey | 456 | 39201 | 0 |
| 8 | 1054 | Sedan | TOYOTA | Camry | 2012 | Automatic | Yes | Hybrid | 2.5 | Front | 4 | 156518 | 4 | 12 | Left wheel | White | 781 | 3607 | 0 |
| 9 | 7945 | Sedan | HYUNDAI | Elantra | 2012 | Automatic | Yes | Petrol | 1.6 | Front | 4 | 165294 | 4 | 4 | Left wheel | Silver | 531 | 16308 | 0 |
| 10 | 15234 | Minivan | MERCEDES-BENZ | Vito | 2007 | Tiptronic | Yes | Diesel | 3.0 | Rear | 6 | 250000 | 4 | 4 | Left wheel | Black | 0 | 30640 | 1 |
| 11 | 2277 | Jeep | LEXUS | RX 450 | 2010 | Automatic | Yes | Hybrid | 3.5 | 4x4 | 6 | 167222 | 4 | 12 | Left wheel | Black | 1399 | 5018 | 0 |
| 12 | 1660 | Sedan | HYUNDAI | Sonata | 2016 | Automatic | Yes | LPG | 2 | Front | 4 | 287140 | 4 | 4 | Left wheel | White | 891 | 18817 | 0 |
| 13 | 15966 | Sedan | FORD | F150 | 2016 | Automatic | Yes | Petrol | 3.5 | Front | 4 | 33543 | 4 | 4 | Left wheel | White | 1493 | 126322 | 0 |
| 14 | 11541 | Coupe | HYUNDAI | Genesis | 2010 | Automatic | Yes | Petrol | 3.8 | Front | 4 | 151977 | 4 | 4 | Left wheel | Blue | 1511 | 16621 | 0 |
| 15 | 1579 | Jeep | TOYOTA | RAV 4 | 2010 | Variator | Yes | Petrol | 2 | 4x4 | 4 | 167300 | 6 | 8 | Left wheel | Blue | 0 | 23207 | 0 |
| 16 | 3011 | Jeep | HYUNDAI | Tucson | 2016 | Automatic | Yes | Diesel | 2 | Front | 4 | 27243 | 4 | 4 | Left wheel | Grey | 891 | 29633 | 0 |
| 17 | 4573 | Jeep | MERCEDES-BENZ | ML 350 | 2009 | Automatic | Yes | Diesel | 3.5 | 4x4 | 6 | 274088 | 4 | 12 | Left wheel | Black | 1624 | 6272 | 0 |
| 18 | 6342 | Jeep | MERCEDES-BENZ | GL 450 | 2006 | Automatic | Yes | LPG | 4.5 | 4x4 | 6 | 181000 | 4 | 6 | Left wheel | Black | 0 | 21000 | 1 |
| 19 | 15558 | Sedan | HYUNDAI | Sonata | 2015 | Automatic | Yes | Petrol | 2 | Front | 4 | 59150 | 4 | 4 | Left wheel | Grey | 765 | 42692 | 0 |

```
df2, freq_category = frequency_encoding(df2, 'Category')
df2, freq_manufacturer = frequency_encoding(df2, 'Manufacturer')
df2, freq_model = frequency_encoding(df2, 'Model')
# Prod. Year
df2, freq_gear_box_type = frequency_encoding(df2, 'Gear box type')
df2, label_leather_interior = label_encoding(df2, 'Leather interior')
df2, freq_fuel_type = frequency_encoding(df2, 'Fuel type')
# Engine volume: quitar el turbo y crear variable aparte
df2, freq_drive_wheels = frequency_encoding(df2, 'Drive wheels')
# Cylinders
df2, freq_mileage = frequency_encoding(df2, 'Mileage') # quitar km
# Doors: cambiar >5 por 4
# Airbags
df2, freq_wheel = frequency_encoding(df2, 'Wheel')
df2, freq_color = frequency_encoding(df2, 'Color')
# Sales Fee: cambiar '-' por '0'
df2.head()
```

| | Id | Category | Manufacturer | Model | Prod. year | Gear box type | Leather interior | Fuel type | Engine volume | Drive wheels | Cylinders | Mileage | Doors | Airbags | Wheel | Color | Sales Fee | price | Turbo |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2680 | 0.287567 | 0.196869 | 0.022567 | 2014 | 0.702832 | 1 | 0.211363 | 2.5 | 0.670907 | 4 | 0.000061 | 4 | 4 | 0.922512 | 0.195951 | 777 | 22433 | 0 |
| 1 | 5960 | 0.453183 | 0.015106 | 0.000428 | 2002 | 0.702832 | 0 | 0.528286 | 1.8 | 0.670907 | 4 | 0.006483 | 4 | 2 | 0.922512 | 0.233380 | 0 | 7500 | 0 |
| 2 | 2185 | 0.287567 | 0.196869 | 0.027521 | 2014 | 0.702832 | 1 | 0.211363 | 2 | 0.670907 | 4 | 0.000122 | 4 | 4 | 0.922512 | 0.233380 | 639 | 27284 | 0 |
| 3 | 15905 | 0.453183 | 0.105315 | 0.000061 | 1992 | 0.096875 | 0 | 0.024524 | 2.6 | 0.118097 | 6 | 0.036817 | 4 | 4 | 0.922512 | 0.006850 | 0 | 3450 | 0 |
| 4 | 15337 | 0.018592 | 0.050028 | 0.022690 | 2015 | 0.702832 | 1 | 0.185065 | 1.5 | 0.670907 | 4 | 0.000061 | 4 | 4 | 0.922512 | 0.261941 | 308 | 26644 | 0 |

## OUTLIERS

```
for col in df2.columns:
    df2[col] = pd.to_numeric(df[col])


# Tratar con outliers
def cuantificaOutliers(dataset):
  for col in dataset.columns:
    q1, q3 = np.percentile(dataset[col],[25,75])
    iqr = q3-q1
    lower_bound = q1 - (1.5*iqr)
    upper_bound = q3 + (1.5*iqr)
    outlier = dataset[(dataset[col]<lower_bound)|(dataset[col]>upper_bound)]
    print(col, ' ', outlier.shape[0], ' ', outlier.shape[0]/dataset.shape[0]*100, '%')

cuantificaOutliers(df2)
```

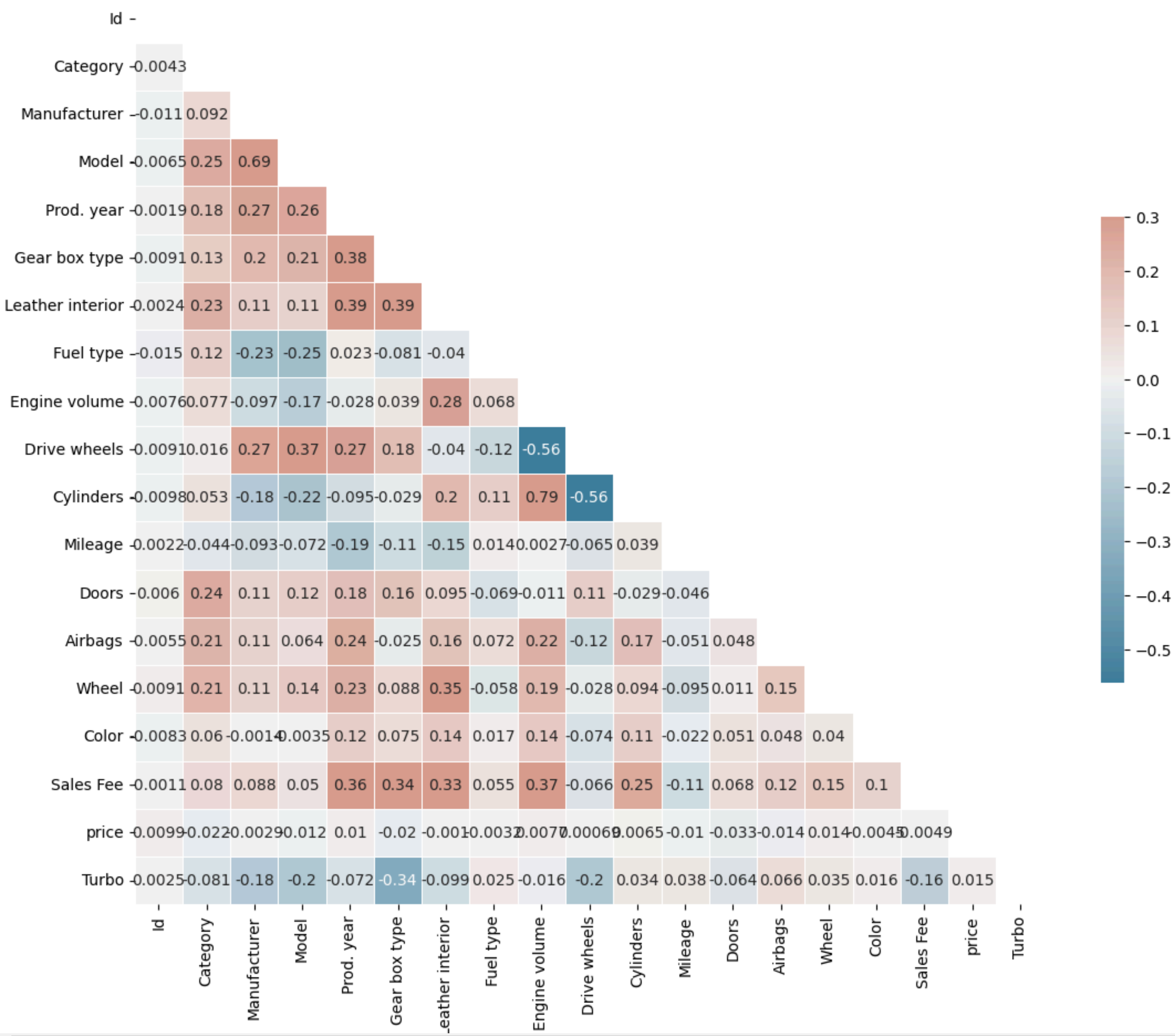
```
Id   0   0.0 %
Category   0   0.0 %
Manufacturer   0   0.0 %
Model   0   0.0 %
Prod. year   824   5.039447128615987 %
Gear box type   0   0.0 %
Leather interior   0   0.0 %
Fuel type   0   0.0 %
Engine volume   1184   7.241147330438505 %
Drive wheels   0   0.0 %
Cylinders   4140   25.31955232095896 %
Mileage   2015   12.323405296312153 %
Doors   763   4.666381261084949 %
Airbags   0   0.0 %
Wheel   1267   7.7487615436364745 %
Color   0   0.0 %
Sales Fee   136   0.831753409577396 %
price   901   5.510366338450248 %
Turbo   1618   9.89541924041343 %
```

## ANÁLISIS DE CORRELACIÓN

```
# Realizar un análisis de correlación
corr = df2.corr(method='pearson')
mask = np.triu(np.ones_like(corr, dtype=bool))
f, ax = plt.subplots(figsize=(11,9))
cmap = sns.diverging_palette(230, 20, as_cmap=True)

plt.tight_layout()
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0, square=True, linewidths=.5, cbar_kws={'shrink':0.5}, annot=True)
```

<Axes: >


Id
Category
Manufacturer
Model
Prod. year
Gear box type
Leather interior
Fuel type
Engine volume
Drive wheels
Cylinders
Mileage
Doors
Airbags
Wheel
Color
Sales Fee
price
Turbo
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5

```
correlations = df2.corr()['price'].abs().sort_values(ascending=False)
print("Correlación con la variable objetivo (Curado):\n", correlations)
```

```
Correlación con la variable objetivo (Curado):
 price             1.000000
Doors             0.032986
Category          0.021632
Gear box type     0.020325
Turbo             0.015388
Wheel             0.013929
Airbags           0.013830
Model             0.012108
Mileage           0.010075
Prod. year        0.010010
Id                0.009915
Engine volume     0.007680
Cylinders         0.006525
Sales Fee         0.004929
Color             0.004539
Fuel type         0.003239
Manufacturer      0.002938
Leather interior  0.000998
Drive wheels      0.000685
Name: price, dtype: float64
```

## VARIABLES

```
df3 = df2
df3 = df3.drop('Model', axis=1)
df3 = df3.drop('Engine volume', axis=1)
df3 = df3.drop('Cylinders', axis=1)
df3 = df3.drop('Sales Fee', axis=1)
df3 = df3.drop('Color', axis=1)
df3 = df3.drop('Mileage', axis=1)
df3 = df3.drop('Fuel type', axis=1)
df3 = df3.drop('Manufacturer', axis=1)
df3 = df3.drop('Leather interior', axis=1)
df3 = df3.drop('Drive wheels', axis=1)
df3.head()
```

| | Id | Category | Prod. year | Gear box type | Doors | Airbags | Wheel | price | Turbo |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2680 | 0.287567 | 2014 | 0.702832 | 4 | 4 | 0.922512 | 22433 | 0 |
| **1** | 5960 | 0.453183 | 2002 | 0.702832 | 4 | 2 | 0.922512 | 7500 | 0 |
| **2** | 2185 | 0.287567 | 2014 | 0.702832 | 4 | 4 | 0.922512 | 27284 | 0 |
| **3** | 15905 | 0.453183 | 1992 | 0.096875 | 4 | 4 | 0.922512 | 3450 | 0 |
| **4** | 15337 | 0.018592 | 2015 | 0.702832 | 4 | 4 | 0.922512 | 26644 | 0 |

```
df4 = df3
y = df4['price']
```

```
x = df4.drop('price', axis=1)
```

## MODELO

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=101)
#model = LogisticRegression(max_iter=100)
#model.fit(x_train,y_train)
#yhat = model.predict(x_test)

# Definir función objetivo para la optimización bayesiana
def xgb_evaluate(max_depth, learning_rate, n_estimators, subsample, colsample_bytree):
    params = {
        'objective': 'reg:squarederror',
        'max_depth': int(max_depth),
        'learning_rate': learning_rate,
        'n_estimators': int(n_estimators),
        'subsample': subsample,
        'colsample_bytree': colsample_bytree,
        'random_state': 42
    }
    model = xgb.XGBRegressor(**params)
    model.fit(x_train, y_train)
    y_val_pred = model.predict(x_test)
    return -root_mean_squared_error(y_test, y_val_pred)


param_bounds = {
    'max_depth': (5, 15),
    'learning_rate': (0.001, 0.05),
    'n_estimators': (1000, 4000),
    'subsample': (0.6, 1.0),
    'colsample_bytree': (0.4, 0.9)
}

# Ejecutar optimización bayesiana
optimizer = BayesianOptimization(f=xgb_evaluate, pbounds=param_bounds, random_state=42, verbose=2)
optimizer.maximize(init_points=10, n_iter=25)

best_params = optimizer.max['params']
best_params['max_depth'] = int(best_params['max_depth'])
best_params['n_estimators'] = int(best_params['n_estimators'])

print("Mejores parámetros encontrados:")
print(best_params)
```

```
|   iter    |  target   | colsam... | learni... | max_depth | n_esti... | subsample |
-------------------------------------------------------------------------------------
| 1         | -4.115e+0 | 0.5873    | 0.04759   | 12.32     | 2.796e+03 | 0.6624    |
| 2         | -4.115e+0 | 0.478     | 0.003846  | 13.66     | 2.803e+03 | 0.8832    |
| 3         | -4.115e+0 | 0.4103    | 0.04853   | 13.32     | 1.637e+03 | 0.6727    |
| 4         | -4.116e+0 | 0.4917    | 0.01591   | 10.25     | 2.296e+03 | 0.7165    |
| 5         | -4.116e+0 | 0.7059    | 0.007835  | 7.921     | 2.099e+03 | 0.7824    |
```

```
| 6         | -4.116e+0 | 0.7926    | 0.01078   | 10.14     | 2.777e+03 | 0.6186    |
| 7         | -4.116e+0 | 0.7038    | 0.009356  | 5.651     | 3.847e+03 | 0.9863    |
| 8         | -4.116e+0 | 0.8042    | 0.01593   | 5.977     | 3.053e+03 | 0.7761    |
| 9         | -4.116e+0 | 0.461     | 0.02526   | 5.344     | 3.728e+03 | 0.7035    |
| 10        | -4.116e+0 | 0.7313    | 0.01627   | 10.2      | 2.64e+03  | 0.6739    |
| 11        | -4.116e+0 | 0.7745    | 0.03179   | 5.16      | 2.798e+03 | 0.6585    |
| 12        | -4.116e+0 | 0.8532    | 0.006162  | 6.473     | 3.776e+03 | 0.9083    |
| 13        | -4.116e+0 | 0.692     | 0.003341  | 5.358     | 1.314e+03 | 0.6933    |
| 14        | -4.116e+0 | 0.6343    | 0.04966   | 14.22     | 2.295e+03 | 0.7262    |
| 15        | -4.116e+0 | 0.8038    | 0.009614  | 8.167     | 1.807e+03 | 0.8904    |
| 16        | -4.115e+0 | 0.6456    | 0.01353   | 12.35     | 2.787e+03 | 0.9192    |
| 17        | -4.116e+0 | 0.4377    | 0.04078   | 9.667     | 1.917e+03 | 0.7195    |
| 18        | -4.115e+0 | 0.4386    | 0.02986   | 13.6      | 2.803e+03 | 0.8724    |
| 19        | -4.115e+0 | 0.5539    | 0.02433   | 13.02     | 2.785e+03 | 0.9035    |
| 20        | -4.115e+0 | 0.7187    | 0.0457    | 10.98     | 2.786e+03 | 0.7871    |
| 21        | -4.116e+0 | 0.7151    | 0.03739   | 12.55     | 2.806e+03 | 0.9389    |
| 22        | -4.116e+0 | 0.8608    | 0.01519   | 10.28     | 2.787e+03 | 0.7208    |
| 23        | -4.116e+0 | 0.7206    | 0.02471   | 13.56     | 2.786e+03 | 0.7206    |
| 24        | -4.116e+0 | 0.8097    | 0.01054   | 10.51     | 2.785e+03 | 0.9654    |
| 25        | -4.116e+0 | 0.8821    | 0.02968   | 12.62     | 2.795e+03 | 0.9889    |
| 26        | -4.115e+0 | 0.715     | 0.02824   | 11.9      | 2.796e+03 | 0.7175    |
| 27        | -4.116e+0 | 0.6831    | 0.003556  | 8.642     | 1.49e+03  | 0.662     |
| 28        | -4.116e+0 | 0.668     | 0.04028   | 13.26     | 2.648e+03 | 0.8185    |
| 29        | -4.115e+0 | 0.5699    | 0.03576   | 12.47     | 2.787e+03 | 0.7985    |
| 30        | -4.116e+0 | 0.6288    | 0.03801   | 11.91     | 2.786e+03 | 0.7665    |
| 31        | -4.116e+0 | 0.897     | 0.04598   | 11.03     | 2.786e+03 | 0.7368    |
| 32        | -4.116e+0 | 0.7994    | 0.01281   | 12.22     | 2.796e+03 | 0.9428    |
| 33        | -4.116e+0 | 0.7013    | 0.004112  | 12.52     | 2.788e+03 | 0.7616    |
| 34        | -4.115e+0 | 0.7253    | 0.04273   | 8.764     | 3.886e+03 | 0.6838    |
| 35        | -4.116e+0 | 0.4369    | 0.02998   | 13.18     | 2.785e+03 | 0.8889    |
=============================================================================
Mejores parámetros encontrados:
{'colsample_bytree': 0.569932594106084, 'learning_rate': 0.03576287390240755, 'max_depth': 12, 'n_estimators': 2787, 'subsample': 0.7985052643584628}
```

```
xgb_reg = xgb.XGBRegressor(objective="reg:squarederror", **best_params)
xgb_reg.fit(x_train, y_train)
predictions = xgb_reg.predict(x_test)
```

## EVALUACIÓN

```
#print('Accuracy: ', metrics.accuracy_score(y_test,predictions))
print((root_mean_squared_error(predictions, y_test)))
```

```
411538.92822731764
```

```
scores = cross_val_score(xgb_reg, x, y, cv=10, scoring='f1_weighted')
print("Scores de cada fold:", scores)
print("Promedio del F1 score:", scores.mean())
```

```
c:\Python312\Lib\site-packages\sklearn\model_selection\_validation.py:1000: UserWarning: Scoring failed. The score on this train-test partition for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "c:\Python312\Lib\site-packages\sklearn\metrics\_scorer.py", line 139, in __call__
    score = scorer._score(
            ^^^^^^^^^^^^^^
```

```
  File "c:\Python312\Lib\site-packages\sklearn\metrics\_scorer.py", line 376, in _score
    return self._sign * self._score_func(y_true, y_pred, **scoring_kwargs)
                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Python312\Lib\site-packages\sklearn\utils\_param_validation.py", line 213, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "c:\Python312\Lib\site-packages\sklearn\metrics\_classification.py", line 1293, in f1_score
    return fbeta_score(
           ^^^^^^^^^^^^
  File "c:\Python312\Lib\site-packages\sklearn\utils\_param_validation.py", line 186, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "c:\Python312\Lib\site-packages\sklearn\metrics\_classification.py", line 1485, in fbeta_score
    _, _, f, _ = precision_recall_fscore_support(
                 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Python312\Lib\site-packages\sklearn\utils\_param_validation.py", line 186, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "c:\Python312\Lib\site-packages\sklearn\metrics\_classification.py", line 1789, in precision_recall_fscore_support
    labels = _check_set_wise_labels(y_true, y_pred, average, labels, pos_label)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Python312\Lib\site-packages\sklearn\metrics\_classification.py", line 1561, in _check_set_wise_labels
    y_type, y_true, y_pred = _check_targets(y_true, y_pred)
                             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Python312\Lib\site-packages\sklearn\metrics\_classification.py", line 112, in _check_targets
    raise ValueError(
ValueError: Classification metrics can't handle a mix of multiclass and continuous targets

  warnings.warn(
c:\Python312\Lib\site-packages\sklearn\model_selection\_validation.py:1000: UserWarning: Scoring failed. The score on this train-test partition for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "c:\Python312\Lib\site-packages\sklearn\metrics\_scorer.py", line 139, in __call__
    score = scorer._score(
            ^^^^^^^^^^^^^^
  File "c:\Python312\Lib\site-packages\sklearn\metrics\_scorer.py", line 376, in _score
    return self._sign * self._score_func(y_true, y_pred, **scoring_kwargs)
                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Python312\Lib\site-packages\sklearn\utils\_param_validation.py", line 213, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "c:\Python312\Lib\site-packages\sklearn\metrics\_classification.py", line 1293, in f1_score
    return fbeta_score(
           ^^^^^^^^^^^^
  File "c:\Python312\Lib\site-packages\sklearn\utils\_param_validation.py", line 186, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "c:\Python312\Lib\site-packages\sklearn\metrics\_classification.py", line 1485, in fbeta_score
    _, _, f, _ = precision_recall_fscore_support(
                 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Python312\Lib\site-packages\sklearn\utils\_param_validation.py", line 186, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "c:\Python312\Lib\site-packages\sklearn\metrics\_classification.py", line 1789, in precision_recall_fscore_support
```

## OUTPUT FILE

```
df_eval = pd.read_csv('../data/Evaluation.csv', sep=';', encoding='latin1')

df_eval['Turbo'] = df_eval['Engine volume'].map(turbo)

df_eval['Sales Fee'] = df_eval['Sales Fee'].map(to_zero)
df_eval['Mileage'] = df_eval['Mileage'].map(mileage_km)
df_eval['Engine volume'] = df_eval['Engine volume'].map(engine_volume)
df_eval['Doors'] = df_eval['Doors'].map(doors)

df_eval['Category'] = df_eval['Category'].map(freq_category).fillna(0)
df_eval['Manufacturer'] = df_eval['Manufacturer'].map(freq_manufacturer)
df_eval['Model'] = df_eval['Model'].map(freq_model)
df_eval['Gear box type'] = df_eval['Gear box type'].map(freq_gear_box_type)
df_eval['Leather interior'] = label_leather_interior.transform(df_eval['Leather interior'])
df_eval['Fuel type'] = df_eval['Fuel type'].map(freq_fuel_type)
df_eval['Drive wheels'] = df_eval['Drive wheels'].map(freq_drive_wheels)
df_eval['Mileage'] = df_eval['Mileage'].map(freq_mileage)
df_eval['Wheel'] = df_eval['Wheel'].map(freq_wheel)
df_eval['Color'] = df_eval['Color'].map(freq_color)

df_eval = df_eval.drop('Model', axis=1)
df_eval = df_eval.drop('Engine volume', axis=1)
df_eval = df_eval.drop('Cylinders', axis=1)
df_eval = df_eval.drop('Sales Fee', axis=1)
df_eval = df_eval.drop('Color', axis=1)
df_eval = df_eval.drop('Mileage', axis=1)
df_eval = df_eval.drop('Fuel type', axis=1)
df_eval = df_eval.drop('Manufacturer', axis=1)
df_eval = df_eval.drop('Leather interior', axis=1)
df_eval = df_eval.drop('Drive wheels', axis=1)

print(df_eval)
```

```
         Id  Category  Prod. year  Gear box type Doors  Airbags     Wheel  \
0     15246  0.453183        2014       0.702832     4        6  0.922512
1      5176  0.453183        2013       0.702832     4       12  0.922512
2      3143  0.287567        2009       0.702832     4        4  0.922512
3      3360  0.287567        2011       0.096875     2        2  0.922512
4      3105  0.027093        2013       0.702832     4       12  0.922512
...     ...       ...         ...            ...   ...      ...       ...
2881  17665  0.453183        2009       0.702832     4       12  0.922512
2882   6554  0.287567        2015       0.702832     4       12  0.922512
2883  18661  0.453183        2014       0.702832     4        0  0.077488
2884   6825  0.453183        2014       0.702832     4        4  0.922512
2885  11266  0.015779        1996       0.096875     4        2  0.922512

      Turbo
```