```
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, freq_leather_interior = frequency_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 | 0.725216 | 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 | 0.274784 | 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 | 0.725216 | 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 | 0.274784 | 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 | 0.725216 | 0.185065 | 1.5 | 0.670907 | 4 | 0.000061 | 4 | 4 | 0.922512 | 0.261941 | 308 | 26644 | 0 |

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


# Interaction terms
df2['Doors_Category'] = df2['Doors'] * df2['Category']
df2['Engine_volume_Cylinders'] = df2['Engine volume'] * df2['Cylinders']
df2['Prod_year_Mileage'] = df2['Prod. year'] * df2['Mileage']

# Additional interaction terms
df2['Doors_ProdYear'] = df2['Doors'] * df2['Prod. year']
df2['Mileage_SalesFee'] = df2['Mileage'] * df2['Sales Fee']
df2['Category_Turbo'] = df2['Category'] * df2['Turbo']

# Polynomial terms
df2['Mileage_Squared'] = df2['Mileage'] ** 2
df2['EngineVolume_Squared'] = df2['Engine volume'] ** 2

# Ratios
df2['EngineVolume_per_Cylinder'] = df2['Engine volume'] / df2['Cylinders']
df2['Mileage_per_Door'] = df2['Mileage'] / df2['Doors']

# Age feature
df2['Car_Age'] = 2024 - df2['Prod. year']

# Interaction with age
df2['Age_Mileage'] = df2['Car_Age'] * df2['Mileage']
df2['Age_SalesFee'] = df2['Car_Age'] * df2['Sales Fee']

# Log transformations (to handle skewness)
df2['Log_Mileage'] = np.log1p(df2['Mileage'])
df2['Log_EngineVolume'] = np.log1p(df2['Engine volume'])
df2['Log_SalesFee'] = np.log1p(df2['Sales Fee'])
```

## OUTLIERS

```
# 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)]
    if (outlier.shape[0] > 0):
      print(col, ' ', outlier.shape[0], ' ', outlier.shape[0]/dataset.shape[0]*100, '%')

cuantificaOutliers(df2)
```

```
Prod. year   824   5.039447128615987 %
Engine volume   1184   7.241147330438505 %
Cylinders   4140   25.31955232095896 %
Mileage   2015   12.323405296312153 %
Doors   763   4.666381261084949 %
Wheel   1267   7.7487615436364745 %
Sales Fee   136   0.831753409577396 %
price   901   5.510366338450248 %
Turbo   1618   9.89541924041343 %
Engine_volume_Cylinders   3426   20.952846920677633 %
Prod_year_Mileage   2014   12.3172894624182 %
Doors_ProdYear   1424   8.708947464986851 %
Mileage_SalesFee   2533   15.491407253378997 %
Category_Turbo   1618   9.89541924041343 %
Mileage_Squared   2750   16.81854320836646 %
EngineVolume_Squared   2373   14.512873830346768 %
EngineVolume_per_Cylinder   198   1.2109351110023852 %
Mileage_per_Door   1994   12.194972784539173 %
Car_Age   824   5.039447128615987 %
Age_Mileage   2240   13.699467922451225 %
Age_SalesFee   548   3.3514769738853896 %
Log_Mileage   2015   12.323405296312153 %
Log_EngineVolume   1089   6.660143110513118 %
```

```
def Modifica_Outliers (dataset,columna):
  q1, q3 = np.percentile(dataset[columna], [25, 75])
  # Calculate the interquartile range
  iqr = q3 - q1
  # Calculate the lower and upper bounds
  lower_limit = q1 - (1.5 * iqr)
  upper_limit = q3 + (1.5 * iqr)

  dataset[columna] = np.where(dataset[columna]>upper_limit,upper_limit,np.where(dataset[columna]<lower_limit,lower_limit,dataset[columna]))
  return (dataset)

Modifica_Outliers(df2,'Engine volume')
Modifica_Outliers(df2,'Prod. year')
Modifica_Outliers(df2,'Mileage')
Modifica_Outliers(df2,'Sales Fee')
Modifica_Outliers(df2,'Engine_volume_Cylinders')
Modifica_Outliers(df2,'Prod_year_Mileage')
Modifica_Outliers(df2,'Doors_ProdYear')
Modifica_Outliers(df2,'Mileage_SalesFee')
Modifica_Outliers(df2,'Age_SalesFee')
Modifica_Outliers(df2,'Log_Mileage')
Modifica_Outliers(df2,'Log_EngineVolume')
Modifica_Outliers(df2,'Car_Age')
```

```
Modifica_Outliers(df2,'Age_Mileage')
Modifica_Outliers(df2,'Mileage_per_Door')
Modifica_Outliers(df2,'Mileage_Squared')
Modifica_Outliers(df2,'EngineVolume_Squared')
Modifica_Outliers(df2,'EngineVolume_per_Cylinder')
cuantificaOutliers(df2)
```

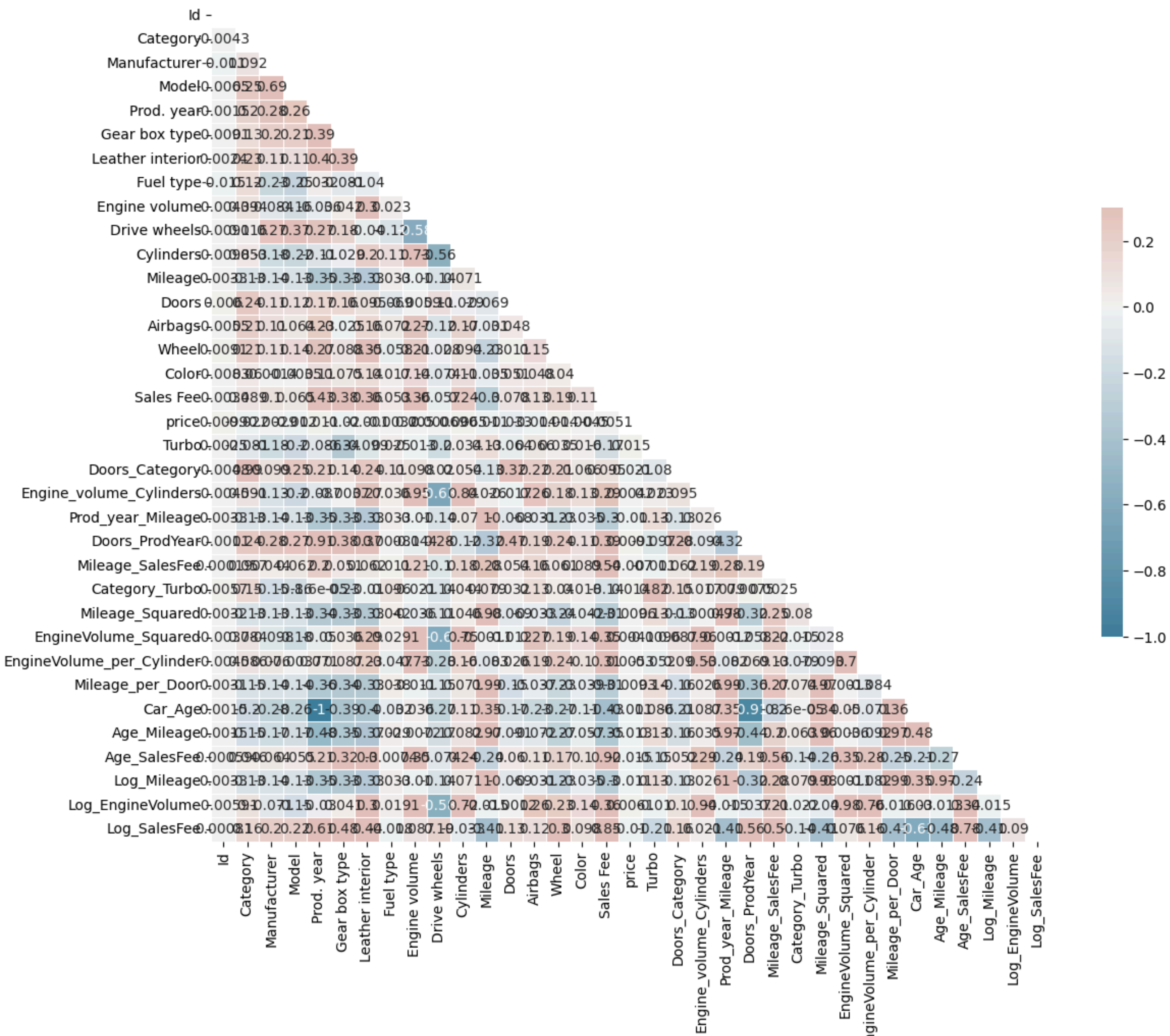
```
Cylinders   4140   25.31955232095896 %
Doors   763   4.666381261084949 %
Wheel   1267   7.7487615436364745 %
price   901   5.510366338450248 %
Turbo   1618   9.89541924041343 %
Category_Turbo   1618   9.89541924041343 %
```

## 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
Doors_Category
Engine_volume_Cylinders
Prod_year_Mileage
Doors_ProdYear
Mileage_SalesFee
Category_Turbo
Mileage_Squared
EngineVolume_Squared
EngineVolume_per_Cylinder
Mileage_per_Door
Car_Age
Age_Mileage
Age_SalesFee
Log_Mileage
Log_EngineVolume
Log_SalesFee
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0

```
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
Doors_Category              0.021222
Gear box type               0.020325
Turbo                       0.015388
Age_SalesFee                0.014557
Category_Turbo              0.014314
Wheel                       0.013929
Airbags                     0.013830
Age_Mileage                 0.013278
Model                       0.012108
Prod. year                  0.010756
Car_Age                     0.010756
Log_Mileage                 0.010523
Mileage                     0.010522
Prod_year_Mileage           0.010499
Log_SalesFee                0.010103
Id                          0.009915
Mileage_Squared             0.009551
Mileage_per_Door            0.009313
Doors_ProdYear              0.009095
Mileage_SalesFee            0.006985
Cylinders                   0.006525
Log_EngineVolume            0.006075
EngineVolume_per_Cylinder   0.005289
Sales Fee                   0.005070
Engine volume               0.005026
Color                       0.004539
Engine_volume_Cylinders     0.004228
EngineVolume_Squared        0.004103
Fuel type                   0.003239
Manufacturer                0.002938
Leather interior            0.000998
Drive wheels                0.000685
Name: price, dtype: float64
```

## VARIABLES

```
df3 = df2
"""
Mileage_Squared             0.009551
Mileage_per_Door            0.009313
Doors_ProdYear              0.009095
Mileage_SalesFee            0.006985
Cylinders                   0.006525
Log_EngineVolume            0.006075
EngineVolume_per_Cylinder   0.005289
```

```
Sales Fee                  0.005070
Engine volume              0.005026
Color                      0.004539
Engine_volume_Cylinders    0.004228
EngineVolume_Squared       0.004103
Fuel type                  0.003239
Manufacturer               0.002938
Leather interior           0.000998
Drive wheels               0.000685
"""
df3 = df3.drop('Mileage_Squared', axis=1)
df3 = df3.drop('Mileage_per_Door', axis=1)
df3 = df3.drop('Doors_ProdYear', axis=1)
df3 = df3.drop('Mileage_SalesFee', axis=1)
df3 = df3.drop('Cylinders', axis=1)
df3 = df3.drop('Log_EngineVolume', axis=1)
df3 = df3.drop('EngineVolume_per_Cylinder', axis=1)
df3 = df3.drop('Sales Fee', axis=1)
df3 = df3.drop('Engine volume', axis=1)
df3 = df3.drop('Color', axis=1)
df3 = df3.drop('Engine_volume_Cylinders', axis=1)
df3 = df3.drop('EngineVolume_Squared', 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 | Model | Prod. year | Gear box type | Mileage | Doors | Airbags | Wheel | price | Turbo | Doors_Category | Prod_year_Mileage | Category_Turbo | Car_Age | Age_Mileage | Age_SalesFee | Log_Mileage | Log_SalesFee |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2680 | 0.287567 | 0.022567 | 2014.0 | 0.702832 | 0.000061 | 4 | 4 | 0.922512 | 22433 | 0 | 1.150266 | 0.123173 | 0.0 | 10.0 | 0.000612 | 7770.0 | 0.000061 | 6.656727 |
| **1** | 5960 | 0.453183 | 0.000428 | 2002.0 | 0.702832 | 0.003272 | 4 | 2 | 0.922512 | 7500 | 0 | 1.812733 | 6.582839 | 0.0 | 22.0 | 0.043055 | 0.0 | 0.003270 | 0.000000 |
| **2** | 2185 | 0.287567 | 0.027521 | 2014.0 | 0.702832 | 0.000122 | 4 | 4 | 0.922512 | 27284 | 0 | 1.150266 | 0.246346 | 0.0 | 10.0 | 0.001223 | 6390.0 | 0.000122 | 6.461468 |
| **3** | 15905 | 0.453183 | 0.000061 | 2000.0 | 0.096875 | 0.003272 | 4 | 4 | 0.922512 | 3450 | 0 | 1.812733 | 6.582839 | 0.0 | 24.0 | 0.043055 | 0.0 | 0.003270 | 0.000000 |
| **4** | 15337 | 0.018592 | 0.022690 | 2015.0 | 0.702832 | 0.000061 | 4 | 4 | 0.922512 | 26644 | 0 | 0.074369 | 0.123234 | 0.0 | 9.0 | 0.000550 | 2772.0 | 0.000061 | 5.733341 |

```
df4 = df3
y = df4['price']
x = df4.drop('price', axis=1)
```

## MODELO

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Separar Dataset en Training y Testing Sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

```python
# Definir función para calcular el RMSE
def root_mean_squared_error(y_true, y_pred):
    return np.sqrt(mean_squared_error(y_true, y_pred))

# Función de evaluación para Random Forest
def random_forest_evaluate(max_depth, n_estimators, max_features, min_samples_split, min_samples_leaf):
    model = RandomForestRegressor(
        max_depth=int(max_depth),
        n_estimators=int(n_estimators),
        max_features=max_features,
        min_samples_split=int(min_samples_split),
        min_samples_leaf=int(min_samples_leaf),
        random_state=42,
        n_jobs=-1  # Usar todos los procesadores disponibles
    )
    model.fit(x_train, y_train)
    y_val_pred = model.predict(x_test)
    return -root_mean_squared_error(y_test, y_val_pred)

# Definir límites para los parámetros de optimización
param_bounds = {
    'max_depth': (5, 20),
    'n_estimators': (100, 1500),
    'max_features': (0.1, 0.9),
    'min_samples_split': (2, 15),
    'min_samples_leaf': (1, 10)
}

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

# Obtener los mejores parámetros
best_params = optimizer.max['params']
best_params['max_depth'] = int(best_params['max_depth'])
best_params['n_estimators'] = int(best_params['n_estimators'])
best_params['min_samples_split'] = int(best_params['min_samples_split'])
best_params['min_samples_leaf'] = int(best_params['min_samples_leaf'])

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

# Inicializar y entrenar el modelo con los mejores parámetros
rf_regressor = RandomForestRegressor(**best_params, random_state=42, n_jobs=-1)
rf_regressor.fit(x_train, y_train)

# Hacer predicciones
y_pred = rf_regressor.predict(x_test)

# Calcular y mostrar el RMSE en el conjunto de prueba
test_rmse = root_mean_squared_error(y_test, y_pred)
print("RMSE en el conjunto de prueba:", test_rmse)
```

```
|   iter    |  target   | max_depth | max_fe... | min_sa... | min_sa... | n_esti... |
-------------------------------------------------------------------------------------
| 1         | -4.601e+0 | 10.62     | 0.8606    | 7.588     | 9.783     | 318.4     |
| 2         | -4.601e+0 | 7.34      | 0.1465    | 8.796     | 9.814     | 1.091e+03 |
| 3         | -4.602e+0 | 5.309     | 0.8759    | 8.492     | 4.76      | 354.6     |
| 4         | -4.601e+0 | 7.751     | 0.3434    | 5.723     | 7.615     | 507.7     |
| 5         | -4.601e+0 | 14.18     | 0.2116    | 3.629     | 6.763     | 738.5     |
| 6         | -4.601e+0 | 16.78     | 0.2597    | 5.628     | 9.701     | 165.0     |
| 7         | -4.601e+0 | 14.11     | 0.2364    | 1.585     | 14.34     | 1.452e+03 |
| 8         | -4.601e+0 | 17.13     | 0.3437    | 1.879     | 10.9      | 716.2     |
| 9         | -4.601e+0 | 6.831     | 0.4961    | 1.309     | 13.82     | 462.3     |
| 10        | -4.601e+0 | 14.94     | 0.3494    | 5.681     | 9.107     | 358.8     |
| 11        | -4.601e+0 | 10.81     | 0.6529    | 3.322     | 12.82     | 953.7     |
| 12        | -4.601e+0 | 18.6      | 0.1843    | 2.325     | 14.03     | 1.179e+03 |
| 13        | -4.601e+0 | 13.76     | 0.1382    | 1.322     | 3.359     | 426.5     |
| 14        | -4.601e+0 | 12.03     | 0.8944    | 9.297     | 7.614     | 541.7     |
| 15        | -4.601e+0 | 10.67     | 0.6233    | 4.493     | 13.13     | 953.6     |
| 16        | -4.601e+0 | 14.78     | 0.3122    | 3.648     | 7.935     | 427.6     |
| 17        | -4.601e+0 | 8.381     | 0.4143    | 2.758     | 2.499     | 427.8     |
| 18        | -4.601e+0 | 13.82     | 0.8548    | 2.538     | 3.134     | 420.7     |
| 19        | -4.601e+0 | 19.85     | 0.7881    | 2.723     | 2.696     | 423.2     |
| 20        | -4.601e+0 | 11.11     | 0.2013    | 2.838     | 10.48     | 719.9     |
| 21        | -4.601e+0 | 16.3      | 0.2501    | 2.104     | 3.092     | 431.5     |
| 22        | -4.601e+0 | 16.66     | 0.5704    | 8.721     | 14.43     | 717.8     |
| 23        | -4.601e+0 | 19.43     | 0.7154    | 2.626     | 6.985     | 723.9     |
| 24        | -4.601e+0 | 14.16     | 0.7941    | 4.295     | 4.056     | 712.7     |
| 25        | -4.601e+0 | 15.61     | 0.4042    | 1.086     | 13.78     | 709.7     |
| 26        | -4.601e+0 | 6.217     | 0.7985    | 2.477     | 10.38     | 710.4     |
| 27        | -4.601e+0 | 19.11     | 0.4934    | 7.16      | 5.081     | 717.8     |
| 28        | -4.601e+0 | 13.7      | 0.8313    | 3.755     | 13.31     | 728.9     |
| 29        | -4.601e+0 | 19.83     | 0.1718    | 1.711     | 6.859     | 707.3     |
| 30        | -4.601e+0 | 19.82     | 0.2964    | 5.022     | 11.78     | 703.6     |
| 31        | -4.601e+0 | 9.733     | 0.3711    | 1.284     | 3.182     | 953.6     |
| 32        | -4.601e+0 | 16.81     | 0.6239    | 1.345     | 7.122     | 959.0     |
| 33        | -4.601e+0 | 7.11      | 0.5769    | 1.518     | 5.978     | 961.0     |
| 34        | -4.601e+0 | 15.59     | 0.3595    | 2.571     | 2.619     | 951.7     |
| 35        | -4.601e+0 | 9.721     | 0.3627    | 1.02      | 5.68      | 944.5     |
=====================================================================================
Mejores parámetros encontrados:
{'max_depth': 13, 'max_features': 0.1382265536174302, 'min_samples_leaf': 1, 'min_samples_split': 3, 'n_estimators': 426}
RMSE en el conjunto de prueba: 460092.9421161285
```

## EVALUACIÓN

```
from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("Root Mean Squared Error (RMSE):", rmse)
print("R^2 Score:", r2)
```

```
Root Mean Squared Error (RMSE): 460092.9421161285
R^2 Score: 4.739949901499951e-05
```

```
from sklearn.model_selection import cross_val_score

#  cross-validation
cv_scores = cross_val_score(rf_regressor, x, y, cv=5, scoring='neg_mean_squared_error')
cv_rmse = np.sqrt(-cv_scores)

print("Cross-Validated RMSE:", cv_rmse.mean())
```

```
Cross-Validated RMSE: 125417.41955628296
```

## 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'] = df_eval['Leather interior'].map(freq_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)

for col in df_eval.columns:
    df_eval[col] = pd.to_numeric(df_eval[col])

# Interaction terms
df_eval['Doors_Category'] = df_eval['Doors'] * df_eval['Category']
df_eval['Engine_volume_Cylinders'] = df_eval['Engine volume'] * df_eval['Cylinders']
df_eval['Prod_year_Mileage'] = df_eval['Prod. year'] * df_eval['Mileage']

# Additional interaction terms
df_eval['Doors_ProdYear'] = df_eval['Doors'] * df_eval['Prod. year']
df_eval['Mileage_SalesFee'] = df_eval['Mileage'] * df_eval['Sales Fee']
df_eval['Category_Turbo'] = df_eval['Category'] * df_eval['Turbo']

# Polynomial terms
df_eval['Mileage_Squared'] = df_eval['Mileage'] ** 2
df_eval['EngineVolume_Squared'] = df_eval['Engine volume'] ** 2

# Ratios
```

```
df_eval['EngineVolume_per_Cylinder'] = df_eval['Engine volume'] / df_eval['Cylinders']
df_eval['Mileage_per_Door'] = df_eval['Mileage'] / df_eval['Doors']

# Age feature
df_eval['Car_Age'] = 2024 - df_eval['Prod. year']

# Interaction with age
df_eval['Age_Mileage'] = df_eval['Car_Age'] * df_eval['Mileage']
df_eval['Age_SalesFee'] = df_eval['Car_Age'] * df_eval['Sales Fee']

# Log transformations (to handle skewness)
df_eval['Log_Mileage'] = np.log1p(df_eval['Mileage'])
df_eval['Log_EngineVolume'] = np.log1p(df_eval['Engine volume'])
df_eval['Log_SalesFee'] = np.log1p(df_eval['Sales Fee'])


df_eval = df_eval.drop('Mileage_Squared', axis=1)
df_eval = df_eval.drop('Mileage_per_Door', axis=1)
df_eval = df_eval.drop('Doors_ProdYear', axis=1)
df_eval = df_eval.drop('Mileage_SalesFee', axis=1)
df_eval = df_eval.drop('Cylinders', axis=1)
df_eval = df_eval.drop('Log_EngineVolume', axis=1)
df_eval = df_eval.drop('EngineVolume_per_Cylinder', axis=1)
df_eval = df_eval.drop('Sales Fee', axis=1)
df_eval = df_eval.drop('Engine volume', axis=1)
df_eval = df_eval.drop('Color', axis=1)
df_eval = df_eval.drop('Engine_volume_Cylinders', axis=1)
df_eval = df_eval.drop('EngineVolume_Squared', 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     Model  Prod. year  Gear box type   Mileage  Doors  \
0     15246  0.453183  0.048560        2014       0.702832  0.001590      4
1      5176  0.453183  0.049477        2013       0.702832  0.000795      4
2      3143  0.287567  0.002324        2009       0.702832       NaN      4
3      3360  0.287567  0.000550        2011       0.096875  0.005321      2
4      3105  0.027093  0.001835        2013       0.702832  0.000306      4
...     ...       ...       ...         ...            ...       ...    ...
2881  17665  0.453183  0.056021        2009       0.702832  0.000245      4
2882   6554  0.287567  0.027521        2015       0.702832       NaN      4
2883  18661  0.453183  0.017369        2014       0.702832  0.003303      4
2884   6825  0.453183  0.000673        2014       0.702832       NaN      4
2885  11266  0.015779  0.011009        1996       0.096875       NaN      4

      Airbags     Wheel  Turbo  Doors_Category  Prod_year_Mileage  \
0           6  0.922512      0        1.812733           3.202495
1          12  0.922512      0        1.812733           1.600453
2           4  0.922512      0        1.150266                NaN
3           2  0.922512      0        0.575133          10.700080
4          12  0.922512      0        0.108373           0.615559
...       ...       ...    ...             ...                ...
2881       12  0.922512      0        1.812733           0.491468
2882       12  0.922512      0        1.150266                NaN
```

```
2883       0  0.077488      0        1.812733           6.651336
2884       4  0.922512      0        1.812733                NaN
2885       2  0.922512      0        0.063115                NaN

      Category_Turbo  Car_Age  Age_Mileage  Age_SalesFee  Log_Mileage  \
0                0.0       10     0.015901          5840     0.001589
1                0.0       11     0.008746          8569     0.000795
2                0.0       15          NaN         17115          NaN
3                0.0       13     0.069170             0     0.005307
4                0.0       11     0.003364           957     0.000306
...              ...      ...          ...           ...          ...
2881             0.0       15     0.003670         11190     0.000245
2882             0.0        9          NaN          8100          NaN
2883             0.0       10     0.033026             0     0.003297
2884             0.0       10          NaN         10530          NaN
2885             0.0       28          NaN             0          NaN
```