+ Código + Texto

```
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

```python
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(df2[col])


# Crear características adicionales basadas en correlaciones y relaciones avanzadas
df2['Mileage_Engine_ratio'] = df2['Mileage'] / (df2['Engine volume'] + 1)
df2['Age'] = 2024 - df2['Prod. year']
df2['Mileage_Age'] = df2['Mileage'] * df2['Age']
df2['Mileage_Engine_Age'] = df2['Mileage'] * df2['Engine volume'] * df2['Age']
df2['Mileage_Age_squared'] = (df2['Mileage'] * df2['Age']) ** 2
df2['log_Mileage'] = np.log1p(df2['Mileage'])
df2['Age_SalesFee'] = df2['Age'] * df2['Sales Fee']
df2['Mileage_Age_Log'] = np.log1p(df2['Mileage_Age'])


# 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 %
Mileage_Engine_ratio   2058   12.586386153752063 %
Age   824   5.039447128615987 %
Mileage_Age   2240   13.699467922451225 %
Mileage_Engine_Age   2150   13.149042871995597 %
Mileage_Age_squared   3023   18.488165861415204 %
log_Mileage   2015   12.323405296312153 %
Age_SalesFee   548   3.3514769738853896 %
Mileage_Age_Log   2240   13.699467922451225 %
```

```
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,'Mileage_Engine_ratio')
Modifica_Outliers(df2,'Age')
Modifica_Outliers(df2,'Mileage_Age')
Modifica_Outliers(df2,'Mileage_Engine_Age')
Modifica_Outliers(df2,'Mileage_Age_squared')
Modifica_Outliers(df2,'log_Mileage')
Modifica_Outliers(df2,'Age_SalesFee')
Modifica_Outliers(df2,'Mileage_Age_Log')
cuantificaOutliers(df2)
```

```
Id   0   0.0 %
Category   0   0.0 %
Manufacturer   0   0.0 %
Model   0   0.0 %
Prod. year   0   0.0 %
Gear box type   0   0.0 %
Leather interior   0   0.0 %
Fuel type   0   0.0 %
Engine volume   0   0.0 %
Drive wheels   0   0.0 %
Cylinders   4140   25.31955232095896 %
Mileage   0   0.0 %
Doors   763   4.666381261084949 %
Airbags   0   0.0 %
Wheel   1267   7.7487615436364745 %
Color   0   0.0 %
Sales Fee   0   0.0 %
```

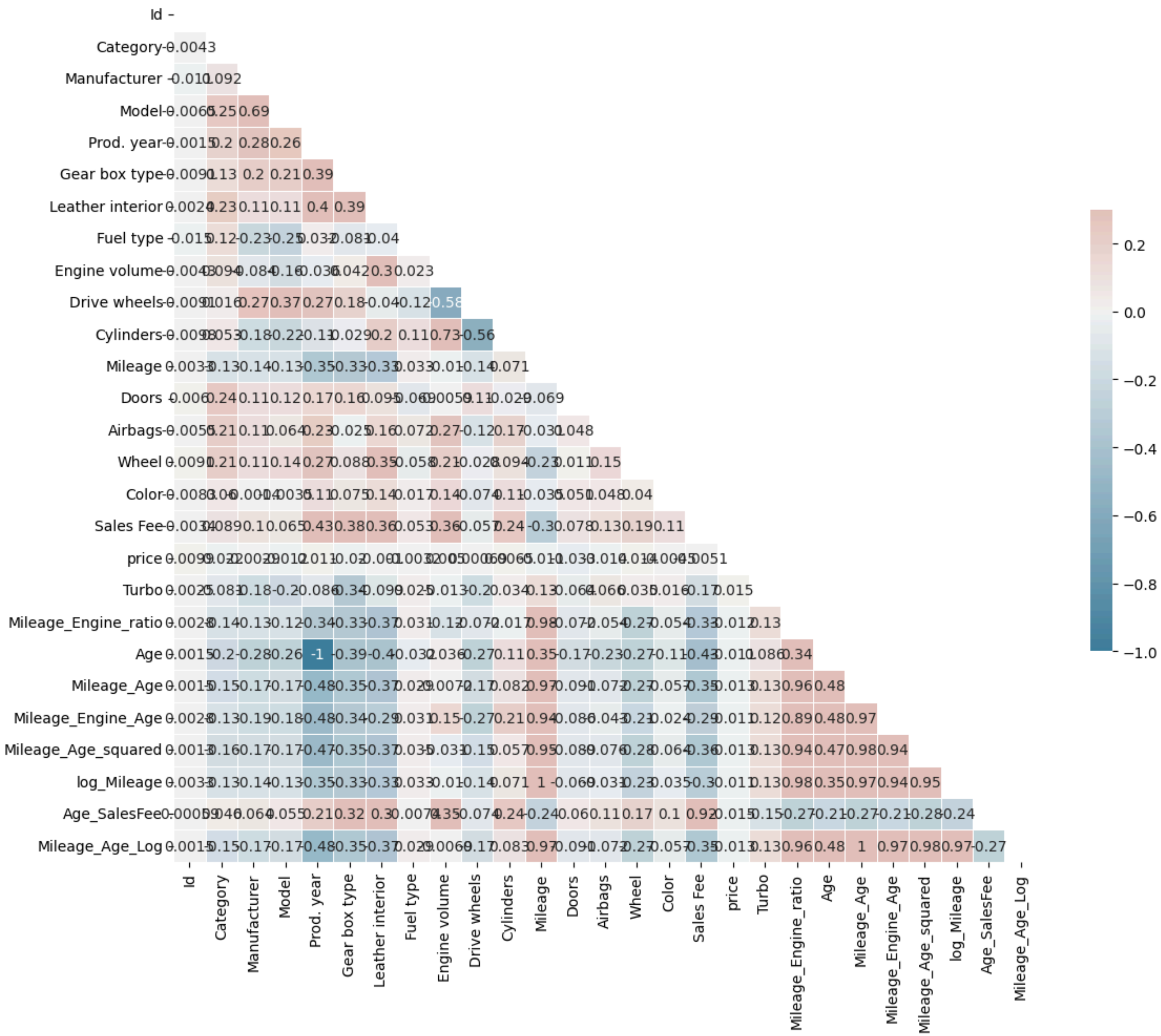
```
price   901   5.510366338450248 %
Turbo   1618   9.89541924041343 %
Mileage_Engine_ratio   0   0.0 %
Age   0   0.0 %
Mileage_Age   0   0.0 %
Mileage_Engine_Age   0   0.0 %
Mileage_Age_squared   0   0.0 %
log_Mileage   0   0.0 %
Age_SalesFee   0   0.0 %
Mileage_Age_Log   0   0.0 %
```

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
Mileage_Engine_ratio
Age
Mileage_Age
Mileage_Engine_Age
Mileage_Age_squared
log_Mileage
Age_SalesFee
Mileage_Age_Log
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
Gear box type         0.020325
Turbo                 0.015388
Age_SalesFee          0.014557
Wheel                 0.013929
Airbags               0.013830
Mileage_Age_Log       0.013287
Mileage_Age           0.013278
Mileage_Age_squared   0.013014
Mileage_Engine_ratio  0.012200
Model                 0.012108
Mileage_Engine_Age    0.011147
Prod. year            0.010756
Age                   0.010756
log_Mileage           0.010523
Mileage               0.010522
Id                    0.009915
Cylinders             0.006525
Sales Fee             0.005070
Engine volume         0.005026
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.head()
```

	Id	Category	Manufacturer	Model	Prod. year	Gear box type	Leather interior	Fuel type	Engine volume	Drive wheels	...	price	Turbo	Mileage_Engine_ratio	Age	Mileage_Age	Mileage_Engine_Age	Mileage_Age_squared	log_Mileage	Age_SalesFee	Mi
0	2680	0.287567	0.196869	0.022567	2014.0	0.702832	1	0.211363	2.5	0.670907	...	22433	0	0.000017	10.0	0.000612	0.001529	3.740342e-07	0.000061	7770.0	
1	5960	0.453183	0.015106	0.000428	2002.0	0.702832	0	0.528286	1.8	0.670907	...	7500	0	0.001028	22.0	0.043055	0.099920	7.853073e-04	0.003270	0.0	
2	2185	0.287567	0.196869	0.027521	2014.0	0.702832	1	0.211363	2.0	0.670907	...	27284	0	0.000041	10.0	0.001223	0.002446	1.496137e-06	0.000122	6390.0	
3	15905	0.453183	0.105315	0.000061	2000.0	0.096875	0	0.024524	2.6	0.118097	...	3450	0	0.001028	24.0	0.043055	0.099920	7.853073e-04	0.003270	0.0	
4	15337	0.018592	0.050028	0.022690	2015.0	0.702832	1	0.185065	1.5	0.670907	...	26644	0	0.000024	9.0	0.000550	0.000826	3.029677e-07	0.000061	2772.0	

5 rows × 27 columns

```
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)

# 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, 15),
    'n_estimators': (100, 1000),
    'max_features': (0.1, 0.9),
    'min_samples_split': (2, 10),
    'min_samples_leaf': (1, 5)
}

# 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 | 8.745     | 0.8606    | 3.928     | 6.789     | 240.4     |
| 2         | -4.601e+0 | 6.56      | 0.1465    | 4.465     | 6.809     | 737.3     |
| 3         | -4.601e+0 | 5.206     | 0.8759    | 4.33      | 3.699     | 263.6     |
| 4         | -4.601e+0 | 6.834     | 0.3434    | 3.099     | 5.456     | 362.1     |
| 5         | -4.601e+0 | 11.12     | 0.2116    | 2.169     | 4.931     | 510.5     |
| 6         | -4.601e+0 | 12.85     | 0.2597    | 3.057     | 6.739     | 141.8     |
| 7         | -4.601e+0 | 11.08     | 0.2364    | 1.26      | 9.591     | 969.1     |
| 8         | -4.601e+0 | 13.08     | 0.3437    | 1.391     | 7.474     | 496.1     |
| 9         | -4.601e+0 | 6.22      | 0.4961    | 1.138     | 9.275     | 332.9     |
| 10        | -4.601e+0 | 11.63     | 0.3494    | 3.08      | 6.374     | 266.4     |
| 11        | -4.601e+0 | 11.05     | 0.2764    | 3.503     | 6.573     | 266.4     |
| 12        | -4.601e+0 | 13.21     | 0.7209    | 3.422     | 5.49      | 501.4     |
| 13        | -4.601e+0 | 5.482     | 0.3468    | 3.855     | 7.704     | 500.3     |
| 14        | -4.601e+0 | 12.97     | 0.5768    | 4.195     | 2.555     | 497.6     |
| 15        | -4.601e+0 | 14.49     | 0.5119    | 1.297     | 9.396     | 502.2     |
| 16        | -4.601e+0 | 14.12     | 0.697     | 4.403     | 8.82      | 273.4     |
| 17        | -4.601e+0 | 14.93     | 0.5921    | 1.257     | 3.375     | 489.9     |
| 18        | -4.601e+0 | 13.22     | 0.8054    | 1.119     | 8.402     | 484.1     |
| 19        | -4.601e+0 | 14.21     | 0.8906    | 3.111     | 2.498     | 477.1     |
| 20        | -4.601e+0 | 5.191     | 0.1048    | 2.573     | 2.536     | 480.5     |
| 21        | -4.601e+0 | 14.96     | 0.3905    | 3.889     | 8.675     | 490.8     |
| 22        | -4.601e+0 | 14.14     | 0.2736    | 1.315     | 8.72      | 474.1     |
| 23        | -4.601e+0 | 14.36     | 0.535     | 4.798     | 3.615     | 465.1     |
| 24        | -4.601e+0 | 9.742     | 0.3381    | 4.027     | 7.597     | 152.6     |
| 25        | -4.601e+0 | 13.29     | 0.3239    | 3.751     | 8.484     | 131.6     |
| 26        | -4.601e+0 | 5.642     | 0.4734    | 3.54      | 2.164     | 134.2     |
| 27        | -4.601e+0 | 9.134     | 0.4944    | 3.035     | 9.422     | 282.0     |
| 28        | -4.601e+0 | 9.8       | 0.6413    | 3.662     | 9.967     | 459.9     |
| 29        | -4.601e+0 | 14.97     | 0.8931    | 4.408     | 4.516     | 485.3     |
| 30        | -4.601e+0 | 14.55     | 0.174     | 3.389     | 9.924     | 120.2     |
| 31        | -4.601e+0 | 13.74     | 0.8323    | 2.105     | 2.122     | 455.7     |
| 32        | -4.601e+0 | 13.59     | 0.365     | 4.982     | 3.845     | 448.1     |
| 33        | -4.601e+0 | 7.404     | 0.258     | 4.607     | 2.044     | 457.0     |
| 34        | -4.601e+0 | 14.98     | 0.635     | 3.547     | 7.143     | 457.8     |
| 35        | -4.601e+0 | 14.65     | 0.181     | 4.932     | 8.258     | 479.1     |
=====================================================================================
Mejores parámetros encontrados:
{'max_depth': 14, 'max_features': 0.5921245624295629, 'min_samples_leaf': 1, 'min_samples_split': 3, 'n_estimators': 489}
RMSE en el conjunto de prueba: 460054.9018297536
```

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): 460054.9018297536
R^2 Score: 0.00021274396444737054
```

```
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: 144337.46606765004
```

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)

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

```
# Crear características adicionales basadas en correlaciones y relaciones avanzadas
df_eval['Mileage_Engine_ratio'] = df_eval['Mileage'] / (df_eval['Engine volume'] + 1)
df_eval['Age'] = 2024 - df_eval['Prod. year']
df_eval['Mileage_Age'] = df_eval['Mileage'] * df_eval['Age']
df_eval['Mileage_Engine_Age'] = df_eval['Mileage'] * df_eval['Engine volume'] * df_eval['Age'
df_eval['Mileage_Age_squared'] = (df_eval['Mileage'] * df_eval['Age']) ** 2
df_eval['log_Mileage'] = np.log1p(df_eval['Mileage'])
df_eval['Age_SalesFee'] = df_eval['Age'] * df_eval['Sales Fee']
df_eval['Mileage_Age_Log'] = np.log1p(df_eval['Mileage_Age'])
```

```
         Id  Category  Manufacturer     Model  Prod. year  Gear box type  \
0     15246  0.453183      0.196869  0.048560        2014       0.702832
1      5176  0.453183      0.192526  0.049477        2013       0.702832
2      3143  0.287567      0.007522  0.002324        2009       0.702832
3      3360  0.287567      0.007522  0.000550        2011       0.096875
4      3105  0.027093      0.105315  0.001835        2013       0.702832
...     ...       ...           ...       ...         ...            ...
2881  17665  0.453183      0.192526  0.056021        2009       0.702832
2882   6554  0.287567      0.196869  0.027521        2015       0.702832
2883  18661  0.453183      0.192526  0.017369        2014       0.702832
2884   6825  0.453183      0.056205  0.000673        2014       0.702832
2885  11266  0.015779      0.056205  0.011009        1996       0.096875

      Leather interior  Fuel type  Engine volume  Drive wheels  ...  \
0                    0   0.528286            1.8      0.670907  ...
1                    1   0.185065            2.5      0.670907  ...
2                    1   0.528286            2.4      0.670907  ...
3                    0   0.528286            3.8      0.210996  ...
4                    1   0.528286            0.0      0.118097  ...
...                ...        ...            ...           ...  ...
2881                 1   0.185065            1.5      0.670907  ...
2882                 1   0.528286            2.4      0.670907  ...
2883                 0   0.185065            1.5      0.670907  ...
2884                 1   0.528286            3.5      0.670907  ...
2885                 0   0.211363            2.5      0.118097  ...

      Sales Fee  Turbo  Mileage_Engine_ratio  Age  Mileage_Age  \
0           584      0              0.000568   10     0.015901
1           779      0              0.000227   11     0.008746
2          1141      0                   NaN   15          NaN
3             0      0              0.001108   13     0.069170
4            87      0              0.000306   11     0.003364
...         ...    ...                   ...  ...          ...
2881        746      0              0.000098   15     0.003670
2882        900      0                   NaN    9          NaN
2883          0      0              0.001321   10     0.033026
2884       1053      0                   NaN   10          NaN
2885          0      0                   NaN   28          NaN

      Mileage_Engine_Age  Mileage_Age_squared  log_Mileage  Age_SalesFee  \
```