Actividad 8 "Modelos descriptivos y predictivos "

Analitica de datos

Licenciatura en Tecnologías de la información

Jesús Santiago Martínez Velarde

Fecha: 20 de febrero 2026

Codigo de alumno:219439844

Semana 5 - Modelos descriptivos y predictivos (caso: Automóviles)

Dataset Automovile_data.csv

1. **Analisis descriptivo**
    - Limpieza de datos.
    - Estadistica descriptiva.
2. **Análisis predictivo**
    - Modelo para predecir price (precio).
    - Evaluación con metricas y lectura de resultados.

Nota: No se busca el "mejor" modelo del mundo, se busca entender el proceso y aprender a interpretar.

In [47]:
```
# Importar las librerias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Modelo predictivo
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
```

In [8]:
```
#Configuración visual
plt.rcParams["figure.figsize"] = (9, 4.5)
plt.rcParams["axes.grid"] = True
```

In [12]:
```
# 2) Cargar el dataset
path = "Automobile_data.csv"
df = pd.read_csv(path)

print("Disemsión (Filas, Columnas)", df.shape)
df.head()
```

```
Disemsión (Filas, Columnas) (205, 26)
```

Out[12]:

g	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	e
3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	
3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	
1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	
2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	
2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	

olumns

3) Vista General del conjunto de datos

Antes de modelar, hay que entender

- ¿Que variables existen?
- ¿Cuales son númericas vs categóricas?
- ¿Hay valores faltantes o anómalos (p. ej.)?
- ¿Cuál es el objetivo? Aquí sera **Price**

In [14]:
```
#Tipos de columnas y conteo de faltantes (incluyendo '?')
df.info()

#Conteo rápido de valores '?' por columna si es que aplica
q_counts = (df == "?").sum().sort_values(ascending=False)
q_counts[q_counts > 0 ]
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          205 non-null    int64  
 1   normalized-losses  205 non-null    object 
 2   make               205 non-null    object 
 3   fuel-type          205 non-null    object 
 4   aspiration         205 non-null    object 
 5   num-of-doors       205 non-null    object 
 6   body-style         205 non-null    object 
 7   drive-wheels       205 non-null    object 
 8   engine-location    205 non-null    object 
 9   wheel-base         205 non-null    float64
 10  length             205 non-null    float64
 11  width              205 non-null    float64
 12  height             205 non-null    float64
 13  curb-weight        205 non-null    int64  
 14  engine-type        205 non-null    object 
 15  num-of-cylinders   205 non-null    object 
 16  engine-size        205 non-null    int64  
 17  fuel-system        205 non-null    object 
 18  bore               205 non-null    object 
 19  stroke             205 non-null    object 
 20  compression-ratio  205 non-null    float64
 21  horsepower         205 non-null    object 
 22  peak-rpm           205 non-null    object 
 23  city-mpg           205 non-null    int64  
 24  highway-mpg        205 non-null    int64  
 25  price              205 non-null    object 
dtypes: float64(5), int64(5), object(16)
memory usage: 41.8+ KB
```

Out[14]:
```
normalized-losses    41
price                 4
stroke                4
bore                  4
peak-rpm              2
num-of-doors          2
horsepower            2
dtype: int64
```

4) Limpieza mínima (Para análisis y modelado)

Este dataset usa ? como *missing value* en algunas columnas.

Vamos a :

1. Remplazar ? por NaN
2. Convertir a numéricas las columnas que deberían de ser numéricas.
3. Confirmar faltantes.

In [15]:
```
# 1) Remplazar '?' por NaN
df_clean = df.replace("?", np.nan).copy()
# Intentar convertir a numéricas donde aplique
# si no se queda igual
for col in df_clean.columns:
    df_clean[col] = pd.to_numeric(df_clean[col], errors="ignore")
```

```
C:\Users\alocal\AppData\Local\Temp\ipykernel_31024\2020369362.py:6: FutureWarning: e
rrors='ignore' is deprecated and will raise in a future version. Use to_numeric with
out passing `errors` and catch exceptions explicitly instead
  df_clean[col] = pd.to_numeric(df_clean[col], errors="ignore")
```

In [16]:
```
# 3) revisar faltantes
missing = df_clean.isna().sum().sort_values(ascending=False)
missing[missing > 0]
```

Out[16]:
```
normalized-losses    41
price                 4
stroke                4
bore                  4
peak-rpm              2
num-of-doors          2
horsepower            2
dtype: int64
```

5) Análisis descriptivo

Estadistica descriptiva Aquí responderemos preguntas como: -¿Cuál es el rango típico de precios? - ¿Qué tal dispersos son los valores? -¿Hay outliers?

In [17]:
```
num_cols = df_clean.select_dtypes(include = [np.number]).columns
df_clean[num_cols].describe().T
```

Out[17]:

	count	mean	std	min	25%	50%	75%	
symboling	205.0	0.834146	1.245307	-2.00	0.00	1.00	2.00	
normalized-losses	164.0	122.000000	35.442168	65.00	94.00	115.00	150.00	25
wheel-base	205.0	98.756585	6.021776	86.60	94.50	97.00	102.40	12
length	205.0	174.049268	12.337289	141.10	166.30	173.20	183.10	20
width	205.0	65.907805	2.145204	60.30	64.10	65.50	66.90	7
height	205.0	53.724878	2.443522	47.80	52.00	54.10	55.50	5
curb-weight	205.0	2555.565854	520.680204	1488.00	2145.00	2414.00	2935.00	406
engine-size	205.0	126.907317	41.642693	61.00	97.00	120.00	141.00	32
bore	201.0	3.329751	0.273539	2.54	3.15	3.31	3.59	
stroke	201.0	3.255423	0.316717	2.07	3.11	3.29	3.41	
compression-ratio	205.0	10.142537	3.972040	7.00	8.60	9.00	9.40	2
horsepower	203.0	104.256158	39.714369	48.00	70.00	95.00	116.00	28
peak-rpm	203.0	5125.369458	479.334560	4150.00	4800.00	5200.00	5500.00	660
city-mpg	205.0	25.219512	6.542142	13.00	19.00	24.00	30.00	4
highway-mpg	205.0	30.751220	6.886443	16.00	25.00	30.00	34.00	5
price	201.0	13207.129353	7947.066342	5118.00	7775.00	10295.00	16500.00	454(

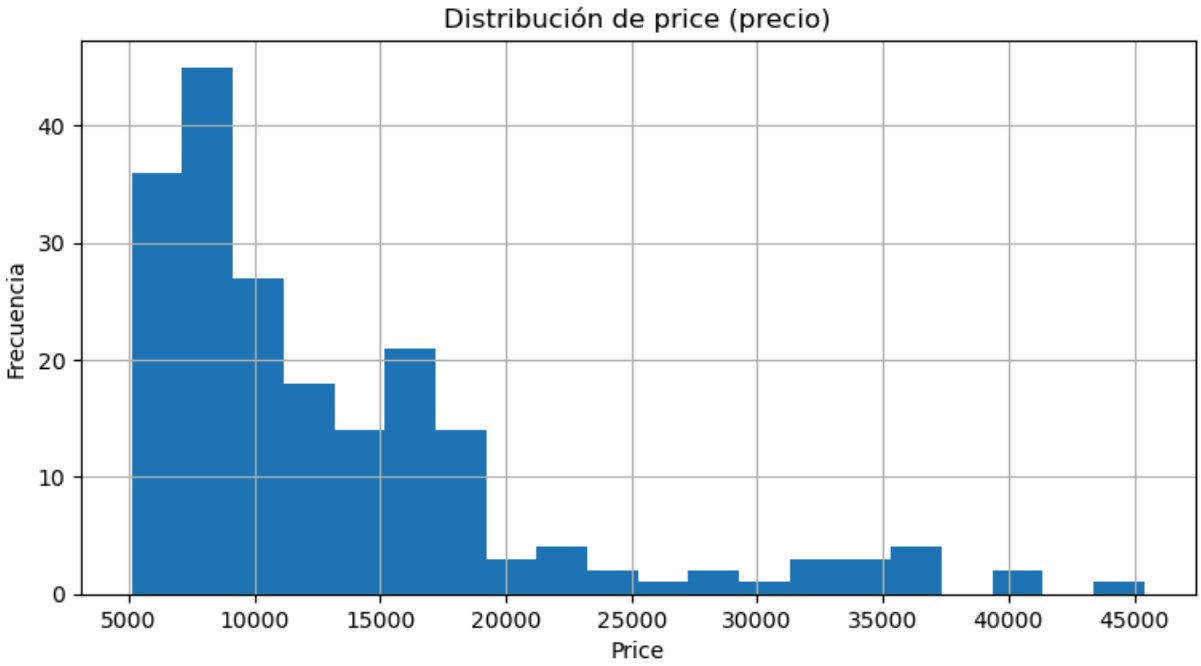
Distribución del precio (price)

- Si el precio esta muy sesgado (cola larga), es comun verlo en autos
- Esto afecta la interpretación y tambien puede afectar el entrenamiento de modelos.

In [19]:
```
#Convertimos price a numérico (por si viniera como object en algunos entornos)
df_clean["price"] = pd.to_numeric(df_clean["price"], errors = "coerce")
```

In [28]:
```
plt.hist(df_clean["price"].dropna(), bins=20)
plt.title("Distribución de price (precio)")
plt.xlabel("Price")
plt.ylabel("Frecuencia")
plt.show()

print("Price: nulos=", df_clean["price"].isna().sum())
```



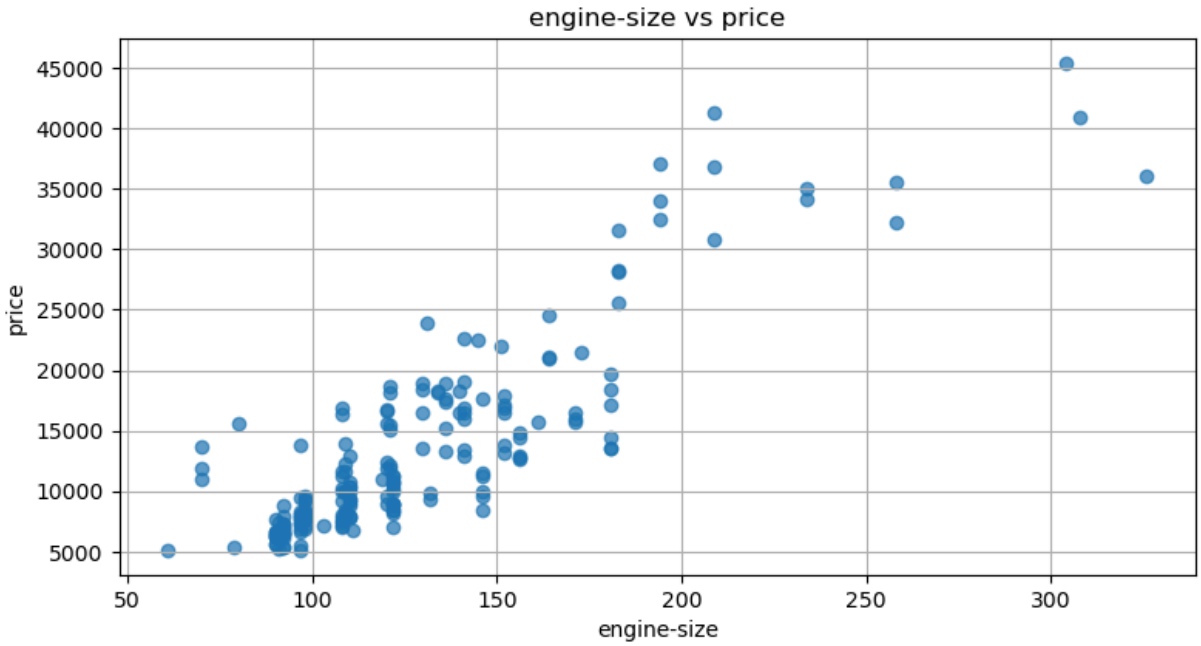

```
Price: nulos= 4
```

Relaciones ej: engine-size vs price

Hipotesis intuitiva: motores más grandes --> autos más caros (no siempre, pero suele correlacionar).

In [31]:
```
plt.scatter(df_clean["engine-size"], df_clean["price"], alpha=0.7)
plt.title("engine-size vs price")
plt.xlabel("engine-size")
plt.ylabel("price")
plt.show()

corr = df_clean[["engine-size", "price"]].corr(numeric_only=True).iloc[0,1]
print("Correlación (engine-size, price):", round(corr, 3))
```




```
Correlación (engine-size, price): 0.872
```

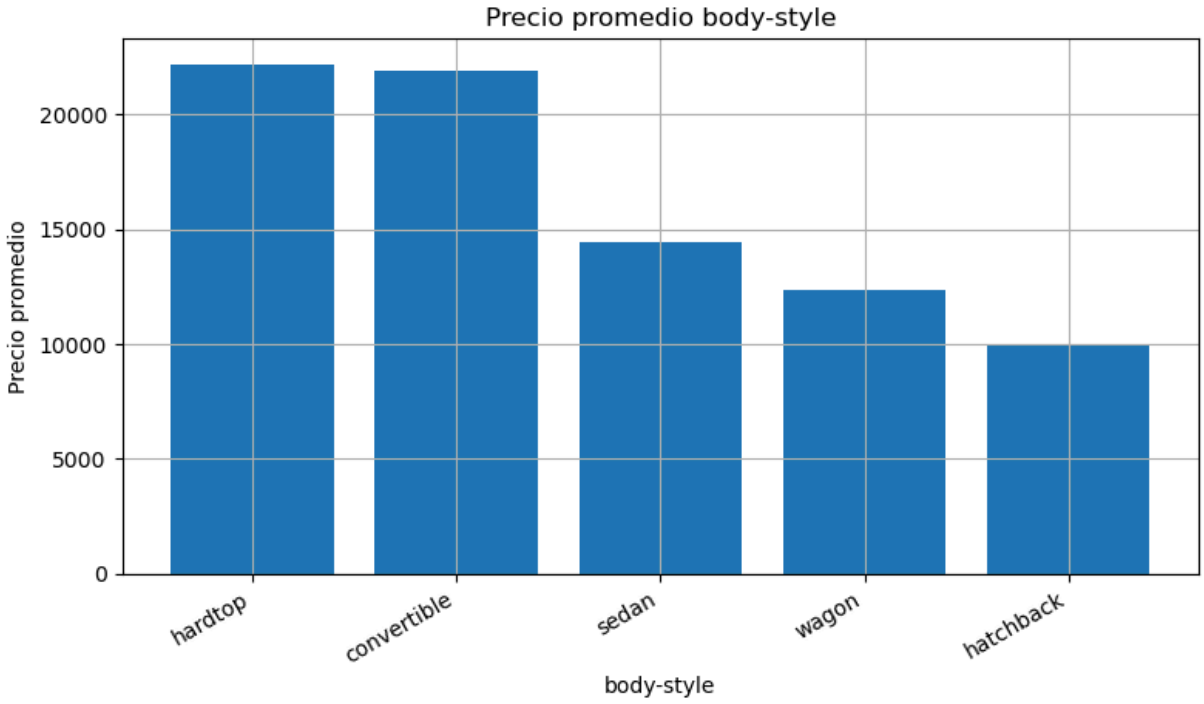
Comparación por categoría: bdy-style y price

Esto es descriptivo: comparar distribuciones por grupos

In [40]:
```
# Precio promedio por tipo de categoría
group = df_clean.groupby("body-style")["price"].mean().sort_values(ascending=False)
group

#Grafica de barras (promedio)

plt.bar(group.index.astype(str), group.values)
plt.title("Precio promedio body-style")
plt.xlabel("body-style")
plt.ylabel("Precio promedio")
plt.xticks(rotation=30, ha="right")
plt.show()
```



Interpretación descriptiva

Cuando presentes resultados descriptivos, usa esta esrtructura:

1. **Hallazgo:** qué observas (p. ej "los sedanes tienen menor precio promedio que los convertibles").
2. **Evidencia:** número/gráfica específica (media, mediana, dispersión, correlación).
3. **Implicación:** Por qué importa (p. ej. segmentación del mercado, estrategía de inventario).
4. **Limitación:** qué no puedes afirmar (correlación causalidad; faltantes; tamaño de muestra).

Análisis predictivo

Cambio de la pregunta:

- **Descriptivo:** "¿Cómo se comportan las variables y cómo se relacionan?"
- **Predictivo:** "Dado un auto con ciertas características, ¿Cuánto costaría?"

**Preparación de datos para el modelado

Usaremos un *pipeline* - imputar faltantes (numéricos y categóticos) -Codificar variables categóticas (one-hot) -entrenar un modelo

In [43]:
```
# Separar X cacateristicas y objetivo
df_model = df_clean.copy()
```

```
# Asegurar que 'price' sea numérica
df_model["price"] = pd.to_numeric(df_model["price"], errors="coerce")

# Quitamos lineas donde el objetivo sea nulo
df_model = df_model.dropna(subset=["price"]).reset_index(drop=True)

X = df_model.drop(columns=["price"])
y = df_model["price"]

# Columnas bumericas y categóricas
num_features = X.select_dtypes(include=[np.number]).columns.tolist()
cat_features = X.select_dtypes(exclude=[np.number]).columns.tolist()

num_features, cat_features, df_model.shape
```

Out[43]:
```
(['symboling',
  'normalized-losses',
  'wheel-base',
  'length',
  'width',
  'height',
  'curb-weight',
  'engine-size',
  'bore',
  'stroke',
  'compression-ratio',
  'horsepower',
  'peak-rpm',
  'city-mpg',
  'highway-mpg'],
 ['make',
  'fuel-type',
  'aspiration',
  'num-of-doors',
  'body-style',
  'drive-wheels',
  'engine-location',
  'engine-type',
  'num-of-cylinders',
  'fuel-system'],
 (201, 26))
```

In [45]:
```
# train/ test split (20/80) para evaluar generalizació
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print("Train: ", X_train.shape, "Test: ", X_test.shape)
```

```
Train:  (160, 25) Test:  (41, 25)
```

In [62]:
```
# Prepocesamiento
numeric_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="median"))
])
```

```
categorical_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("onehot", OneHotEncoder(handle_unknown="ignore"))
])

preprocess = ColumnTransformer(
    transformers=[
        ("num", numeric_transformer, num_features),
        ("cat", categorical_transformer, cat_features),
    ]
)

linereg = Pipeline(steps=[
    ("preprocess", preprocess),
    ("model", LinearRegression())
])

# Entrenar
linereg.fit(X_train, y_train)

# Predecir
pred_lin = linereg.predict(X_test)

# Métricas
mae_rf = mean_absolute_error(y_test, pred_lin)
rmse_rf = np.sqrt(mean_squared_error(y_test, pred_lin))
r2_rf = r2_score(y_test, pred_lin)

print("=== Regresion Lineal ===")
print("MAE: ", round(mae_rf,2))
print("RMSE: ", round(rmse_rf,2))
print("R^2: ", round(r2_rf,3))
```

```
=== Regresion Lineal ===
MAE:  1972.15
RMSE:  3252.12
R^2:  0.914
```

In [63]:
```
# Prepocesamiento
numeric_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="median"))
])

categorical_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("onehot", OneHotEncoder(handle_unknown="ignore"))
])

preprocess = ColumnTransformer(
    transformers=[
        ("num", numeric_transformer, num_features),
        ("cat", categorical_transformer, cat_features),
    ]
)
```

```
linereg = Pipeline(steps=[
    ("preprocess", preprocess),
    ("model", LinearRegression())
])

# Entrenar
linereg.fit(X_train, y_train)

# Predecir
pred_lin = linereg.predict(X_test)

# Métricas
mae = mean_absolute_error(y_test, pred_lin)
rmse = np.sqrt(mean_squared_error(y_test, pred_lin))
r2 = r2_score(y_test, pred_lin)

print("=== Regresion Lineal ===")
print("MAE: ", round(mae,2))
print("RMSE: ", round(rmse,2))
print("R^2: ", round(r2,3))
```

```
=== Regresion Lineal ===
MAE:  1972.15
RMSE:  3252.12
R^2:  0.914
```

In [65]:
```
# Modelo random forest
rf = Pipeline(steps=[
    ("preprocess", preprocess),
    ("model", RandomForestRegressor(
        n_estimators=400,
        random_state=42,
        n_jobs= -1
    ))
])

rf.fit(X_train, y_train)
pred_rf = rf.predict(X_test)

mae_rf = mean_absolute_error(y_test, pred_rf)
rmse_rf = np.sqrt(mean_squared_error(y_test, pred_rf))
r2_rf = r2_score(y_test, pred_rf)

print("=== Random Forest ===")
print("MAE: ", round(mae_rf,2))
print("RMSE: ", round(rmse_rf,2))
print("R^2: ", round(r2_rf,3))
```

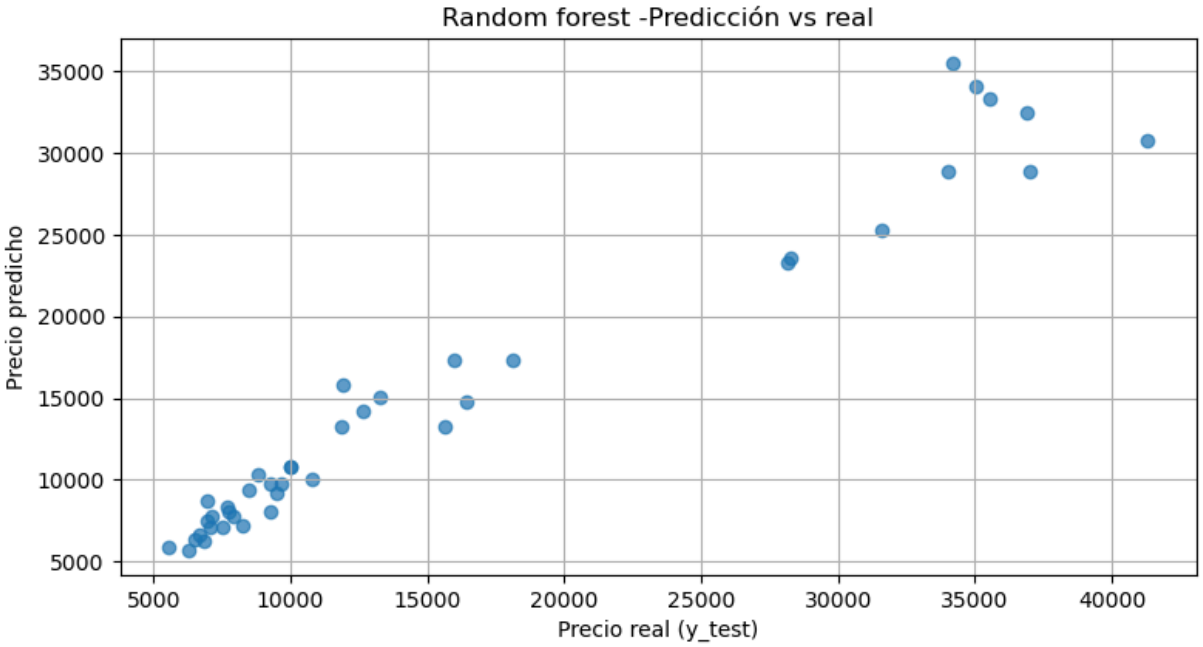
```
=== Random Forest ===
MAE:  1899.76
RMSE:  2985.57
R^2:  0.927
```

Interpretacion de metricas

- **MAE** (Mean absolue error): error promedio en unidades de precio.
- **RMSE** Penaliza más errores grandes, util cuando te preocupan errores muy grandes
- **R^2** Proporción de la varianza explicada (1.0 es perfecto).
  - En datos reales, valores moderados pueden ser razobales.

Lo importante: comparar modelos y decidir segun el contexto

In [66]:
```
# Visualizacion predicción vs real
plt.scatter(y_test, pred_rf, alpha=0.7)
plt.title("Random forest -Predicción vs real")
plt.xlabel("Precio real (y_test)")
plt.ylabel("Precio predicho")
plt.show()
```



¿Qué variables importan más?

Los arboles permiten estimar la importancia de caracteristicas.

One-hot-encoding, una sola columna categorica se convierte en muchas, asi que interpretemos a nivel general

In [79]:
```
# Recuperar los nombres de features despues del procesamiento para interpretar impo

# Ajustar el preprocesamiento por separado
preprocess_fitted = rf.named_steps["preprocess"]

# Nombres numéricos
num_names = num_features

# Nombres categoricos
ohe = preprocess_fitted.named_transformers_["cat"].named_steps["onehot"]
```

```
cat_names = ohe.get_feature_names_out(cat_features).tolist()

feature_names = num_names + cat_names

importances = rf.named_steps["model"].feature_importances_

imp = (
    pd.Series(importances, index=feature_names)
    .sort_values(ascending=False)
    .head(15)
)
print("Top 15 características más importantes para el modelo:")
imp
```

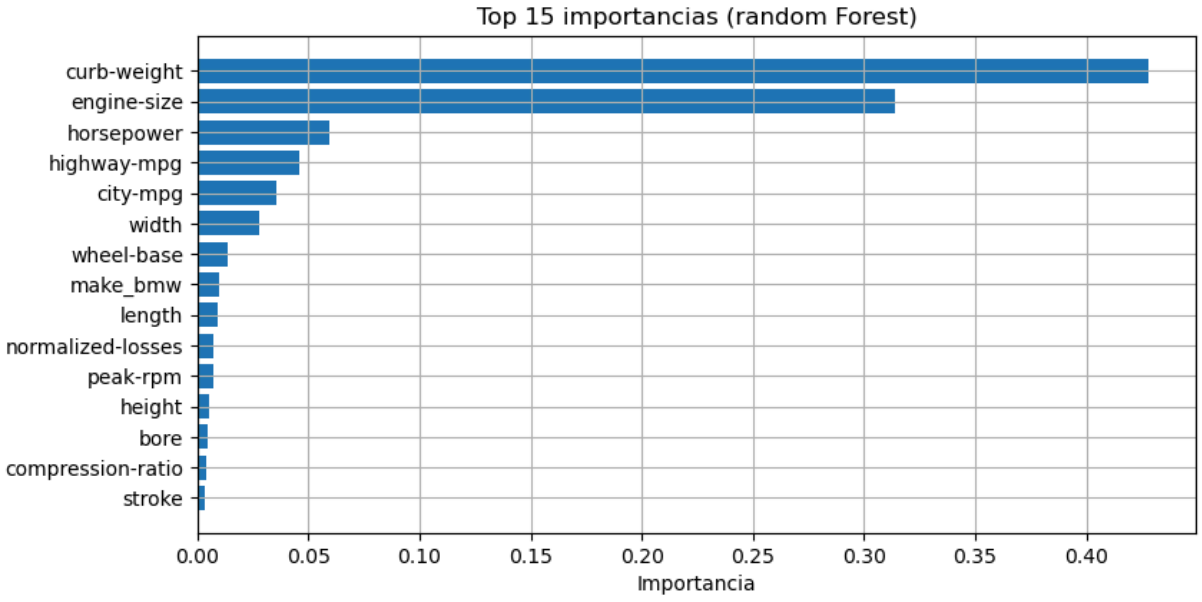
```
Top 15 características más importantes para el modelo:
```

Out[79]:

```
curb-weight          0.427894
engine-size          0.313985
horsepower           0.059449
highway-mpg          0.045753
city-mpg             0.035813
width                0.027695
wheel-base           0.013553
make_bmw             0.009898
length               0.009286
normalized-losses    0.007222
peak-rpm             0.007091
height               0.005450
bore                 0.004518
compression-ratio    0.003951
stroke               0.003260
dtype: float64
```

In [80]:

```
plt.barh(imp.index[::-1], imp.values[::-1])
plt.title("Top 15 importancias (random Forest)")
plt.xlabel("Importancia")
plt.show()
```



1. Grafico de precio promedio por categoria

In [82]:
```
""" # --- GRÁFICO DE PRECIO PROMEDIO POR CATEGORÍA ---
import seaborn as sns

# Ajustamos el tamaño del lienzo
plt.figure(figsize=(10, 6))

# Creamos el gráfico de barras usando los datos de 'group'
# x = el índice de 'group' (los nombres de las carrocerías)
# y = los valores de 'group' (los precios promedio)
sns.barplot(x=group.index, y=group.values, palette='magma')

# Títulos y etiquetas para que se vea profesional
plt.title('Precio Promedio de Autos por Tipo de Carrocería', fontsize=15, fontweigh
plt.xlabel('Tipo de Carrocería (Body Style)', fontsize=12)
plt.ylabel('Precio Promedio', fontsize=12)

# Rotamos los nombres un poco por si son muy largos
plt.xticks(rotation=45)

# Mostramos el gráfico
plt.tight_layout()
plt.show() """
```

Out[82]:
```
" # --- GRÁFICO DE PRECIO PROMEDIO POR CATEGORÍA ---\nimport seaborn as sns\n\n# A
justamos el tamaño del lienzo\nplt.figure(figsize=(10, 6))\n\n# Creamos el gráfico
de barras usando los datos de 'group'\n# x = el índice de 'group' (los nombres de
las carrocerías)\n# y = los valores de 'group' (los precios promedio)\nsns.barplot
(x=group.index, y=group.values, palette='magma')\n\n# Títulos y etiquetas para que
se vea profesional\nplt.title('Precio Promedio de Autos por Tipo de Carrocería', f
ontsize=15, fontweight='bold')\nplt.xlabel('Tipo de Carrocería (Body Style)', font
size=12)\nplt.ylabel('Precio Promedio', fontsize=12)\n\n# Rotamos los nombres un p
oco por si son muy largos\nplt.xticks(rotation=45)\n\n# Mostramos el gráfico\nplt.
tight_layout()\nplt.show() "
```

In [ ]: